PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04L 9/30 — A1

(11) International Publication Number: WO 99/04531

(43) International Publication Date: 28 January 1999 (28.01.99)

(21) International Application Number: PCT/US98/14892

(22) International Filing Date: 17 July 1998 (17.07.98)

(30) Priority Data:
08/896,993 18 July 1997 (18.07.97) US

(71) Applicant: APPLE COMPUTER, INC. [US/US]; 1 Infinite Loop, M/S: 39–PAT, Cupertino, CA 95014 (US).

(72) Inventors: CRANDALL, Richard, E.; 3754 S.E. Knight Street, Portland, OR 97202 (US). GARST, Blaine; 3307 Bay Court, Belmont, CA 94002 (US).

(74) Agents: HECKER, Gary, A. et al.; Hecker & Harriman, Suite 2300, 1925 Century Park East, Los Angeles, CA 90067 (US).

(81) Designated States: CA, JP, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: METHOD AND APPARATUS FOR FAST ELLIPTICAL ENCRYPTION WITH DIRECT EMBEDDING

(57) Abstract

The present invention takes advantage of a quadratic–only ambiguity for x–coordinates in elliptic curve algebra as a means for encrypting plaintext directly onto elliptic curves. The encrypting of plaintext directly onto elliptic curves if refered to herein as "direct embedding". When performing direct embedding, actual plaintext is embedded as a "+" or "–" x–coordinate. The sender specifies using an extra bit whether + or – is used so that the receiver can decrypt appropriately. In operation their are two public initial x–coordinates such that two points P_1^+ and P_1^- lie respectively on two curves E^+ and E^-. A parcel of text x_{text} is selected that is no more than q bits in length. The curve (E^+ or E^-) that contains x_{text} is determined. A random number r is chosen and used to generate a coordinate x_q using the public key of a receiving party. An elliptic add operation is used with the coordinate x_q and the parcel of text to generate a message coordinate x_m. A clue x_c is generated using the random number and the point P from the appropriate curve E+/–. The sign that holds for x_{text} is determined and called g. The message coordinate m_m, the clue x_c, and the sign g are sent as a triple to the receiving party. The receiving party uses the clue x_c and its private key to generate coordinate x_q. Using the sign g and coordinate x_q, the text can be recovered.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

METHOD AND APPARATUS FOR FAST ELLIPTICAL ENCRYPTION WITH DIRECT EMBEDDING

This is a continuation-in-part of United States Patent Application 08/758,688 which is a continuation of United States Patent Application 484,264 (now issued as U. S. Patent Number 5,581,616, which is a continuation in part of United States Patent Application 08/167,408 filed December 14, 1993 which is a continuation of Unites States Patent Application 07/955,479 filed October 2, 1992 (now issued as U. S. Patent 5,271,061) which is a continuation of United States Application Serial Number 07/761,276 filed September 17, 1991 (now issued as U. S. Patent Number 5,159,632).

BACKGROUND OF THE PRESENT INVENTION

1. FIELD OF THE INVENTION

This invention relates to the field of cryptographic systems.

2. BACKGROUND ART

A cryptographic system is a system for sending a message from a sender to a receiver over a medium so that the message is "secure", that is, so that only the intended receiver can recover the message. A cryptographic system converts a message, referred to as "plaintext" into an encrypted format,

known as "ciphertext." The encryption is accomplished by manipulating or transforming the message using a "cipher key" or keys. The receiver "decrypts" the message, that is, converts it from ciphertext to plaintext, by reversing the manipulation or transformation process using the cipher key or keys. So long as only the sender and receiver have knowledge of the cipher key, such an encrypted transmission is secure.

A "classical" cryptosystem is a cryptosystem in which the enciphering information can be used to determine the deciphering information. To provide security, a classical cryptosystem requires that the enciphering key be kept secret and provided to users of the system over secure channels. Secure channels, such as secret couriers, secure telephone transmission lines, or the like, are often impractical and expensive.

A system that eliminates the difficulties of exchanging a secure enciphering key is known as "public key encryption." By definition, a public key cryptosystem has the property that someone who knows only how to encipher a message cannot use the enciphering key to find the deciphering key without a prohibitively lengthy computation. An enciphering function is chosen so that once an enciphering key is known, the enciphering function is relatively easy to compute. However, the inverse of the encrypting transformation function is difficult, or computationally infeasible, to

compute. Such a function is referred to as a "one way function" or as a "trap door function." In a public key cryptosystem, certain information relating to the keys is public. This information can be, and often is, published or transmitted in a non-secure manner. Also, certain information relating to the keys is private. This information may be distributed over a secure channel to protect its privacy, (or may be created by a local user to ensure privacy).

A block diagram of a typical public key cryptographic system is illustrated in Figure 1. A sender represented by the blocks within dashed line 100 sends a plaintext message Ptxt to a receiver, represented by the blocks within dashed line 115. The plaintext message is encrypted into a ciphertext message C, transmitted over some transmission medium and decoded by the receiver 115 to recreate the plaintext message Ptxt.

The sender 100 includes a cryptographic device 101, a secure key generator 102 and a key source 103. The key source 103 is connected to the secure key generator 102 through line 104. The secure key generator 102 is coupled to the cryptographic device 101 through line 105. The cryptographic device provides a ciphertext output C on line 106. The secure key generator 102 provides a key output on line 107. This output is provided, along with the ciphertext message 106, to transmitter receiver 109. The transmitter

receiver 109 may be, for example, a computer transmitting device such as a modem or it may be a device for transmitting radio frequency transmission signals. The transmitter receiver 109 outputs the secure key and the ciphertext message on an insecure channel 110 to the receiver's transmitter receiver 111.

The receiver 115 also includes a cryptographic device 116, a secure key generator 117 and a key source 118. The key source 118 is coupled to the secure key generator 117 on line 119. The secure key generator 117 is coupled to the cryptographic device 116 on line 120. The cryptographic device 116 is coupled to the transmitter receiver 111 through line 121. The secure key generator 117 is coupled to the transmitter receiver 111 on lines 122 and 123.

In operation, the sender 100 has a plaintext message Ptxt to send to the receiver 115. Both the sender 100 and the receiver 115 have cryptographic devices 101 and 116, respectively, that use the same encryption scheme. There are a number of suitable cryptosystems that can be implemented in the cryptographic devices. For example, they may implement the Data Encryption Standard (DES) or some other suitable encryption scheme.

Sender and receiver also have secure key generators 102 and 117, respectively. These secure key generators implement any one of several well

known public key exchange schemes. These schemes, which will be described in detail below, include the Diffie-Hellman scheme, the RSA scheme, the Massey-Omura scheme, and the ElGamal scheme.

The sender 100 uses key source 103, which may be a random number generator, to generate a private key. The private key is provided to the secure key generator 102 and is used to generate an encryption key e_K. The encryption key e_K is transmitted on lines 105 to the cryptographic device and is used to encrypt the plaintext message Ptxt to generate a ciphertext message C provided on line 106 to the transmitter receiver 109. The secure key generator 102 also transmits the information used to convert to the secure key from key source 103 to the encryption key e_K. This information can be transmitted over an insecure channel, because it is impractical to recreate the encryption key from this information without knowing the private key.

The receiver 115 uses key source 118 to generate a private and secure key 119. This private key 119 is used in the secure key generator 117 along with the key generating information provided by the sender 100 to generate a deciphering key D_K. This deciphering key D_K is provided on line 120 to the cryptographic device 116 where it is used to decrypt the ciphertext message and reproduce the original plaintext message.

The Diffie-Hellman Scheme

A scheme for public key exchange is presented in Diffie and Hellman, "New Directions in Cryptography," IEEE Trans. Inform. Theory, vol. IT-22, pp. 644-654, Nov. 1976 (The "DH" scheme). The DH scheme describes a public key system based on the discrete exponential and logarithmic functions. If "q" is a prime number and "a" is a primitive element, then X and Y are in a 1:1 correspondence for $1 \leq X, Y \leq (q-1)$ where $Y = a^X \mod q$, and $X = \log_a Y$ over the finite field. The first discrete exponential function is easily evaluated for a given a and X, and is used to compute the public key Y. The security of the Diffie-Hellman system relies on the fact that no general, fast algorithms are known for solving the discrete logarithm function $X = \log_a Y$ given X and Y.

In a Diffie-Hellman system, a directory of public keys is published or otherwise made available to the public. A given public key is dependent on its associated private key, known only to a user. However, it is not feasible to determine the private key from the public key. For example, a sender has a public key, referred to as "*myPub*". A receiver has a public key, referred to here as "*theirPub*". The sender also has a private key, referred to here as "*myPri*". Similarly, the receiver has a private key, referred to here as "*theirPri*".

There are a number of elements that are publicly known in a public key system. In the case of the Diffie-Hellman system, these elements include a prime number p and a primitive element g. p and g are both publicly known. Public keys are then generated by raising g to the private key power (mod p). For example, a sender's public key $myPub$ is generated by the following equation:

$$myPub = g^{myPri} \pmod{p} \qquad \text{Equation (1)}$$

Similarly, the receiver's public key is generated by the equation:

$$theirPub = g^{theirPri} \pmod{p} \qquad \text{Equation (2)}$$

Public keys are easily created using exponentiation and modulo arithmetic. As noted previously, public keys are easily obtainable by the public. They are published and distributed. They may also be transmitted over non-secure channels. Even though the public keys are known, it is very difficult to calculate the private keys by the inverse function because of the difficulty in solving the discrete log problem.

Figure 2 illustrates a flow chart that is an example of a key exchange using a Diffie-Hellman type system. At step 201, a prime number p is chosen.

This prime number p is public. Next, at step 202, a primitive root g is chosen. This number g is also publicly known. At step 203 an enciphering key e_K is generated, the receiver's public key (*theirPub*) is raised to the power of the sender's private key (*myPri*). That is:

$$(theirPub)^{myPri} \pmod{p} \qquad \text{Equation (3)}$$

We have already defined *theirPub* equal to $g^{theirPri} \pmod{p}$. Therefore Equation 3 can be given by:

$$(g^{theirPri})^{myPri} \pmod{p} \qquad \text{Equation (4)}$$

This value is the enciphering key e_K that is used to encipher the plaintext message and create a ciphertext message. The particular method for enciphering or encrypting the message may be any one of several well known methods. Whichever encrypting message is used, the cipher key is the value calculated in Equation 4. The ciphertext message is then sent to the receiver at step 204.

At step 205, the receiver generates a deciphering key D_K by raising the public key of the sender (*myPub*) to the private key of the receiver (*theirPri*) as follows:

$$D_K = (myPub)^{theirPri} \pmod{p} \qquad \text{Equation (5)}$$

From Equation 1, $myPub$ is equal to $g^{myPri} \pmod{p}$. Therefore:

$$D_K = (g^{myPri})^{theirPri} \pmod{p} \qquad \text{Equation (6)}$$

Since $(g^A)^B$ is equal to $(g^B)^A$, the encipher key e_K and the deciphering key D_K are the same key. These keys are referred to as a "one-time pad." A one-time pad is a key used in enciphering and deciphering a message.

The receiver simply executes the inverse of the transformation algorithm or encryption scheme using the deciphering key to recover the plaintext message at step 206. Because both the sender and receiver must use their private keys for generating the enciphering key, no other users are able to read or decipher the ciphertext message. Note that step 205 can be performed prior to or contemporaneously with any of steps 201-204.

RSA

Another public key cryptosystem is proposed in Rivest, Shamir and Adelman, "On Digital Signatures and Public Key Cryptosystems," Commun.

Ass. Comput. Mach., vol. 21, pp. 120-126, Feb. 1978 (The "RSA" scheme). The RSA scheme is based on the fact that it is easy to generate two very large prime numbers and multiply them together, but it is much more difficult to factor the result, that is, to determine the very large prime numbers from their product. The product can therefore be made public as part of the enciphering key without compromising the prime numbers that effectively constitute the deciphering key.

In the RSA scheme a key generation algorithm is used to select two large prime numbers p and q and multiply them to obtain $n = pq$. The numbers p and q can be hundreds of decimal digits in length. Then Euler's function is computed as $\phi(n) = (p - 1)(q - 1)$. ($\phi(n)$ is the number of integers between 1 and n that have no common factor with n). $\phi(n)$ has the property that for any integer a between 0 and $n - 1$ and any integer k, $a^{k\phi(n) + 1} = a \pmod{n}$.

A random number E is then chosen between 1 and $\phi(n) - 1$ and which has no common factors with $\phi(n)$. The random number E is the enciphering key and is public. This then allows $D = E^{-1} \pmod{\phi(n)}$ to be calculated easily using an extended version of Euclid's algorithm for computing the greatest common divisor of two numbers. D is the deciphering key and is kept secret.

The information (E, n) is made public as the enciphering key and is used to transform unenciphered, plaintext messages into ciphertext messages as follows: a message is first represented as a sequence of integers each between 0 and $n - 1$. Let P denote such an integer. Then the corresponding ciphertext integer is given by the relation $C = P^E \pmod{n}$. The information (D, n) is used as the deciphering key to recover the plaintext from the ciphertext via $P = C^D \pmod{n}$. These are inverse transformations because $C^D = P^{ED} = P^{k\phi(n) + 1} = P$.

MASSEY-OMURA

The Massey-Omura cryptosystem is described in U.S. Patent Number 4,567,600. In the Massey cryptosystem, a finite field F_q is selected. The field F_q is fixed and is a publicly known field. A sender and a receiver each select a random integer e between 0 and $q-1$ so that the greatest common denominator G.C.D. $(e, q-1) = 1$. The user then computes its inverse $D = e^{-1} \pmod{q-1}$ using the Euclidean algorithm. Therefore, $De = 1 \pmod{q-1}$.

The Massey-Omura cryptosystem requires that three messages be sent to achieve a secure transmission. Sender A sends message P to receiver B. Sender A calculates random number e_A and receiver B calculates random number e_B. The sender first sends the receiver the element P^{e_A}. The receiver

is unable to recover P since the receiver does not know e_A. Instead, the receiver raises the element to his own private key e_B and sends a second message $P^{e}{}_{A}{}^{e}{}_{B}$ back to the sender. The sender then removes the effect of e_A by raising the element to the $D_{A\text{-th}}$ power and returns $P^{e}{}_{B}$ to the receiver B. The receiver B can read this message by raising the element to the $D_{B\text{-th}}$ power.

ELGAMAL CRYPTOSYSTEM

The ElGamal public key cryptosystem utilizes a publicly known finite field F_q and an element g of F^*_q. Each user randomly chooses an integer a in the range $0 > a > q-1$. The integer a is the private deciphering key. The public enciphering key is the element g^a of F^*_q. To send a message represented by P to a user A, an integer K is randomly chosen. A pair of elements of F_q, namely (g^K, Pg^{aK}) are sent to A. The plaintext message Ptxt is encrypted with the key g^{aK}. The value g^K is a "clue" to the receiver for determining the plaintext message Ptxt. However, this clue can only be used by someone who knows the secure deciphering key "a". The receiver A, who knows "a", recovers the message P from this pair by raising the first element g^K to the a-th power, forming $(g^K)^a$ and dividing the result into the second element.

ELLIPTIC CURVES

Another form of public key cryptosystem is referred to as an "elliptic curve" cryptosystem. An elliptic curve cryptosystem is based on points on an elliptic curve *E* defined over a finite field *F*. Elliptic curve cryptosystems rely for security on the difficulty in solving the discrete logarithm problem. An advantage of an elliptic curve cryptosystem is there is more flexibility in choosing an elliptic curve than in choosing a finite field. Nevertheless, elliptic curve cryptosystems have not been widely used in computer-based public key exchange systems due to their computational intensiveness. Computer-based elliptic curve cryptosystems are slow compared to other computer public key exchange systems. Elliptic curve cryptosystems are described in "A Course in Number Theory and Cryptography" (Koblitz, 1987, Springer-Verlag, New York).

To date, elliptic curve schemes have been used for key exchange and for authentication. However, there has not been a suitable scheme proposed for using elliptic curve algebra as an encryption scheme itself.

SUMMARY OF THE INVENTION

The present invention provides means for encrypting plaintext directly as points on elliptic curves. This direct embedding using elliptic curve algebra avoids an intermediate encryptor stage, such as a so-called DES stage. The ease of embedding in this invention is a result of choosing elliptic curve parameterizations over a field F_p, where p is a prime number such that $p = 2q - C = 3 \pmod 4$. The integers q and C are chosen such that p be prime, with C (possibly negative) being suitably small in magnitude so that fast arithmetic can be performed.

The ability to treat plaintext directly as points on one of two related curves is a consequence of choosing a prime p such that $p = 3 \pmod 4$. The process of choosing which curve contains the plaintext point is referred to herein as "direct embedding." Direct embedding avoids the non-deterministic algorithm of Koblitz and other, typically complicated approaches.

There are two modes of operation, the first mode being used by the second as a preliminary step. Assume there are two elliptic curves denoted E^+ and its twist E^-, initial points on these curves P_1^+ and P_1^-, public key points for both curves theirPub$^+$, theirPub$^-$ and ourPub$^+$, ourPub$^-$ derived by elliptic

multiplication respectively from the initial points and private keys *theirPri* and *ourPri*. A parcel of plaintext x_{text} is selected that falls in the range 0 to *p-1*. It is determined whether E^+ or E^- contains the point with *x*-coordinate x_{text}. A random number *r* is chosen and used to generate a new coordinate x_q on that curve by elliptic curve multiplication of the appropriate public point theirPub$^-$. Assume an *elliptic_add* operation that can compute the *x*-coordinates of both the addition and subtraction of two points on an elliptic curve, but that does not distinguish which result corresponds to which operation. An *elliptic_add* operation of the point x_{text} and x_q is performed, and one of the two results is chosen to generate the encrypted message point x_m (e.g., x_m is either $x_{text} + x_q$ or $x_{text} - x_q$). The inverse *elliptic_add* operation is performed upon x_{text} and x_q to determine which of the results reverse the operation ($x_m + x_q$ or $x_m - x_q$ will reveal x_{text}), and the choice is denoted as *g*. A clue x_c is formed by elliptic multiplication of the random number *r* and the appropriate initial public point $P^{\pm}$. The triplet (x_m, *g*, x_c) is sent to the receiver. The receiving party computes x_q by elliptic multiplication of x_c by *theirPri*, and computes *elliptic_add* on x_m and x_q , and uses *g* to select the result x_m. The *x*-coordinate of x_m is the original parcel of plaintext.

The second mode of operation reduces the size of each encrypted parcel by establishing a synchronized random number generator between the sender and receiver by using, for example, the first mode to transmit two random

numbers r and s. The first synchronization clue is formed by successive elliptic multiplication of theirPub$^{\pm}$ by *ourPri* and r by the sending party, and ourPub by *theirPri*$^{\pm}$ and r by the receiving party. Successive clues x_{clue_n} are formed by choosing the first result of the *elliptic_add* operation upon the clue $x_{clue_{n-1}}$ and the point formed by elliptic multiplication of the initial point P1$^{\pm}$ by s. Each parcel of plaintext is then encrypted by first determining which curve contains the point x_{text}, and the first result of *elliptic_add* of x_{text} and x_{clue_n} forming x_m, and again noting as g the position of x_m in the results of an *elliptic_add* upon x_m and x_{clue_n}. The pair (x_m, g) is sent to the receiver. The receiver determines which curve x_m belongs upon and performs *elliptic_add* upon x_m and x_{clue_n}, and uses g to select the original message parcel x_m from the two results.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of a prior art public key exchange system.

Figure 2 is a flow diagram of a prior art public key exchange transaction.

Figure 3 is a flow diagram illustrating the key exchange of the present invention.

Figure 4 is a block diagram of a computer system on which the present invention may be implemented.

Figure 5 is a diagram illustrating the shift and add operations for performing mod *p* arithmetic using Mersenne primes.

Figure 6 is a diagram illustrating the operations for performing mod *p* arithmetic using Fermat numbers.

Figure 7 is a diagram illustrating the operations for performing mod *p* arithmetic using fast class numbers.

Figure 8 is a block diagram of the present invention.

Figure 9 is a flow diagram illustrating the operation of one embodiment of the present invention.

Figure 10 is a flow diagram illustrating the generation of a digital signature using the present invention.

Figure 11 is a flow diagram illustrating the authentication of a digital signature in the present invention.

Figure 12 illustrates a block diagram for implementing the digital signature scheme of the present invention.

Figure 13 is a flow diagram of encrypting a plaintext message using direct embedding.

Figure 14 is a flow diagram of decrypting the encrypted message of Figure 13.

Figure 15 is a flow diagram of encrypting a plaintext message using expansionless direct embedding.

Figure 16 is a flow diagram of decrypting the encrypted message of Figure 15.

DETAILED DESCRIPTION OF THE INVENTION

An elliptic curve encryption scheme is described. In the following description, numerous specific details, such as number of bits, execution time, etc., are set forth in detail to provide a more thorough description of the present invention. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without these specific details. In other instances, well known features have not been described in detail so as not to obscure the present invention.

A disadvantage of prior art computer-implemented elliptic curve encryption schemes is they are unsatisfactorily slow compared to other prior art computer-implemented encryption schemes. The modulo arithmetic and elliptic algebra operations required in a prior art elliptic curve cryptosystem require that divisions be performed. Divisions increase computer CPU (central processing unit) computational overhead. CPU's can perform addition and multiplication operations more quickly, and in fewer processing steps, than division operations. Therefore, prior art elliptic curve cryptosystems have not been previously practical or desirable as compared to other prior art cryptosystems, such as Diffie-Hellman and RSA schemes.

The present invention provides methods and apparatus for implementing an elliptic curve cryptosystem for public key exchange that does not require explicit division operations. The advantages of the preferred embodiment of the present invention are achieved by implementing fast classes of numbers, inversionless parameterization, and FFT multiply mod operations.

Elliptic Curve Algebra

The elliptic curve used with the present invention is comprised of points $(x,y) \varepsilon F_{p^k} X F_{p^k}$ satisfying:

$$y^2 = x^3 + cx^2 + ax + b \qquad \text{Equation (7a)}$$

or

$$-y^2 = x^3 + cx^2 + ax + b \qquad \text{Equation (7b)}$$

together with a "point at infinity" A.

The case where $b = 0$ and $a = 1$ is known as the "Montgomery parameterization" and will later be used for purposes of illustration:

$$\pm y^2 = x^3 + c\,x^2 + x \qquad \text{Equation (7c)}$$

and the case where c=0 is known as the Weierstrass parameterization, and will also be used for purposes of illustration:

$$\pm y^2 = x^3 + ax + b \qquad \text{Equation (7d)}$$

Sender ("our") and recipient ("their") private keys are assumed to be integers, denoted:

$$ourPri, theirPri \in Z$$

Next, parameters are established for both sender and recipient. The parameters are:

q, so that $p = 2^q - C$ is a fast class number (q is the "bit-depth"). The value q is a publicly known value.

p and k, so that F_{p^k} will be the field, and where prime p and integer k are publicly known.

$(x_1, y_1) \in F_{p^k}$, the initial x-coordinate, which is publicly known.

$a, b, c \in F_{p^k}$, integer curve defining parameters, all publicly known.

The present invention uses an operation referred to as "elliptic multiplication" and represented by the symbol "°". The operation of elliptic multiplication is well known in the literature, and, for purposes of this patent, will be illustrated using the Weierstrass parameterization of Equation 7d as follows. Similar rules are obtained for other parameterizations, such as the Montgomery parameterization.

An initial point (X_1, Y_1) on the curve of Equation 7d is defined. For the set of integers n, expression $n \circ (X_1, Y_1)$ denotes the point (X_n, Y_n) obtained via the following relations, known as adding and doubling rules.

$$X_{n+1} = ((Y_n - Y_1)/(X_n - X_1))^2 - X_1 - X_n \qquad \text{Equation (8)}$$

$$Y_{n+1} = -Y_1 + ((Y_n - Y_1)/(X_n - X_1))(X_1 - X_{n+1}) \qquad \text{Equation (9)}$$

When $(X_1, Y_1) = (X_n, Y_n)$, the doubling relations to be used are:

$$X_{n+1} = ((3X_1^2 + a)/2Y_1)^2 - 2X_1; \qquad \text{Equation (10)}$$

$$Y_{n+1} = -Y_1 + ((3X_1^2 + a)/2Y_1)(X_1 - X_{n+1}) \qquad \text{Equation (11)}$$

Because arithmetic is performed over the field F_{p^k}, all operations are to be performed (mod p). In particular, the division operation in equations 8 to 11 involve inversions (mod p).

Parameterizations other than Weierstrass are easily formulated.

Elliptic Curve Public Key Exchange

It is necessary that both sender and recipient use the same set of such parameters. Both sender and recipient generate a shared secret pad, as a particular x-coordinate on the elliptic curve.

In the following description, the terms "our" and "our end" refer to the sender. The terms "their" and "their end" refer to the receiver. This convention is used because the key exchange of the present invention may be accomplished between one or more senders and one or more receivers. Thus, "our" and "our end" and "their" and "their end" refers to one or more senders and receivers, respectively.

The public key exchange of the elliptic curve cryptosystem of the present invention is illustrated in the flow diagram of Figure 3.

Step 301- At our end, a public key is computed: ourPub $\varepsilon F_{pk} \times F_{pk}$

$$\text{ourPub} = (ourPri) \circ (x_1, y_1) \qquad \text{Equation (12)}$$

Step 302- At their end, a public key is computed: theirPub $\varepsilon F_{pk} \times F_{pk}$

$$\text{theirPub} = (theirPri) \circ (x_1, y_1) \qquad \text{Equation (13)}$$

Step 303- The two public keys ourPub and theirPub are published, and therefore known to all users.

Step 304- The shared pad is computed at our end: ourPad $\varepsilon F_{pk} \times F_{pk}$

$$\text{ourPad} = (ourPri) \circ (\text{theirPub}) = (ourPri) \circ (theirPri) \circ (x_1, y_1)$$

Equation (14)

Step 305- The shared pad is computed at their end: theirPad $\varepsilon F_{pk} \times F_{pk}$

$$\text{theirPad} = (theirPri) \circ (\text{ourPub}) = (theirPri) \circ (ourPri) \circ (x_1, y_1)$$

Equation (15)

The points on an elliptic curve form an abelian group under the adding and doubling operations above. Therefore, the order of operation of equations 14 and 15 can be changed without affecting the result of the equations. Therefore:

$$\text{ourPad} = (ourPri) \circ (theirPri) \circ (x_1, y_1) = (theirPri) \circ (ourPri) \circ (x_1, y_1)$$
$$= \text{theirPad} \qquad \text{Equation (16)}$$

Since both the sender and receiver use the same pad, the message encrypted by the sender can be decrypted by the recipient. (Note that step 305 can be executed prior to or contemporaneously with any of steps 301-304).

At step 306, the sender encrypts plaintext message Ptxt using **ourPad**, and transmits ciphertext message C to the receiver. At step 307, the receiver decrypts ciphertext message C to recover plaintext message Ptxt, using **theirPad**.

Fast Class Numbers

Elliptic curve cryptosystems make use of modulo arithmetic to determine certain parameters, such as public keys, one time pads, etc. The use of modulo arithmetic serves the dual purpose of limiting the number of bits in the results of equations to some fixed number, and providing security. The discrete log problem is asymmetrical in part because of the use of modulo arithmetic. A disadvantage of modulo arithmetic is the need to perform division operations. The solution to a modulo operation is the remainder

when a number is divided by a fixed number. For example, 12 mod 5 is equal to 2. (5 divides into 12 twice with a remainder of 2, the remainder 2 is the solution). Therefore, modulo arithmetic requires division operations.

Special fast classes of numbers are used in the present invention to optimize the modulo arithmetic required in the enciphering and deciphering process by eliminating the need for division operations. The class of numbers used in the present invention is generally described by the form $2^q - C$ where C is an odd number and is relatively small, (e.g. no longer than the length of a computer word), and where $C = 1 \pmod 4$.

When a number is of this form, modulo arithmetic can be accomplished using shifts, trivial multiplies, and adds only, eliminating the need for divisions. One subset of this fast class is known as "Mersenne" primes, and are of the form $2^q - 1$. Another class that can be used with the present invention are known as "Fermat" numbers of the form $2^q + 1$, where q is equal to 2^m. Fermat numbers may be prime or not prime in the present invention.

The present invention utilizes elliptic curve algebra over a finite field F_{p^k} where $p = 2^q - C$ and p is a fast class number. Note that the expression

$2^q - C$ does not result in a prime number for all values of q and C. For example, when q is equal to 4, and C is equal to 1, $2^q - C$ is equal to 15, not a prime. However, when q has a value of 2, 3, or 5, and C = 1, the expression $2^q - C$ generates the prime numbers 3, 7, and 31.

The present invention implements elliptic curves over a finite field F_{p^k} where p is $2^q - C$ is an element of a fast class of numbers. When practiced on a computer using binary representations of data, the use of fast class numbers allows the (mod p) operations to be accomplished using only shifts and adds. By contrast, the use of "slow" numbers requires that time consuming division operations be executed to perform (mod p) arithmetic. The following examples illustrate the advantage of fast class number (mod p) arithmetic.

Example 1: base 10 (mod p) division

Consider the 32 bit digital number n, where n = 11101101111010111100011100110101 (In base 10 this number is 3,991,652,149).

Now consider n (mod p) where p is equal to 127. The expression n mod 127 can be calculated by division as follows:

```
                31430331
          ┌──────────
127       │ 3991652149
          │  381
          │   181
          │   127
          │    546
          │    508
          │     385
          │     381
          │      42
          │       0
          │      421
          │      381
          │       404
          │       381
          │        239
          │        127
          │        112
```

The remainder 112 is the solution to n mod 127.

Example 2: Mersenne Prime (mod p) Arithmetic

In the present invention, when p is a Mersenne prime where $p = 2^q - 1$, the (mod p) arithmetic can be accomplished using only shifts and adds, with no division required. Consider again n (mod p) where n is 3,991,652,149 and p is 127. When p is 127, q is equal to 7, from $p = 2^q - 1$; $127 = 2^7 - 1 = 128 - 1 = 127$.

The (mod p) arithmetic can be accomplished by using the binary form of n, namely 11101101111010111100011100110101. Referring to Figure 5, the shifts and adds are accomplished by first latching the q least significant bits (LSB's) 501 of n, namely 0110101. The q LSB's 502 of the remaining digits, namely 0001110, are then added to q digits 501, resulting in sum 503 (1000011). The next q LSB's 504 of n, (0101111), are added to sum 503, generating sum 505, (1110010). Bits 506 of n (1101111) are added to sum 505, to result in sum 507, (11100001).

The remaining bits 508 (1110), even though fewer in number than q bits, are added to sum 507 to generate sum 509 (11101111). This sum has greater than q bits. Therefore, the first q bits 510 (1101111) are summed with the next q bits 511 (in this case, the single bit 1), to generate sum 512 (1110000). This sum, having q or fewer bits, is the solution to n (mod p). $1110000 = 2^6 + 2^5 + 2^4 = 64 + 32 + 16 = 112$.

Thus, the solution 112 to n mod 127 is determined using only shifts and adds when an elliptic curve over a field of Mersenne primes is used. The use of Mersenne primes in conjunction with elliptic curve cryptosystems eliminates explicit divisions.

Example 3: Fermat Number (mod p) Arithmetic

In the present invention, when p is a Fermat number where $p = 2^q + 1$, the (mod p) arithmetic can be accomplished using only shifts, adds, and subtracts (a negative add), with no division required. Consider again n (mod p) where n is 3,991,652,149 and where p is now 257. When p is 257, q is equal to 8, from $p = 2^q + 1$; $257 = 2^8 + 1 = 256 + 1 = 257$.

The (mod p) arithmetic can be accomplished by using the binary form of n, namely 11101101111010111100011100110101. Referring to Figure 6, the shifts and adds are accomplished by first latching the q (8) least significant bits (LSB's) 601 (00110101). The next q LSB's 602 of the remaining digits, namely 11000111, are to be subtracted from q digits 601. To accomplish this, the 1's complement of bits 602 is generated and a 1 is added to the MSB side to indicate a negative number, resulting in bits 602' (100111000). This negative number 602' is added to bits 601 to generate result 603 (101101101). The next q LSB's 604 of n, (11101011), are added to sum 603, generating result 605, (1001011000). Bits 606 of n (11101101) are to be subtracted from result 605. Therefore, the 1's complement of bits 606 is generated and a negative sign bit of one is added on the MSB side to generate bits 606' (100010010). Bits 606' is added to result 605, to generate sum 607, (1101101010).

Sum 607 has more than q bits so the q LSB's are latched as bits 608 (01101010). The next q bits (in this case, only two bits, 11) are added to bits 608, generating sum 610 (01101101). This sum, having q or fewer bits, is the solution to n (mod p). $01101101 = 2^6 + 2^5 + 2^3 + 2^2 + 2^0 = 64 + 32 + 8 + 4 + 1 =$ 109.

Example 4: Fast Class mod arithmetic

In the present invention, when p is a number of the class $p = 2^q - C$, where C is and odd number and is relatively small, (e.g. no greater than the length of a digital word), the (mod p) arithmetic can be accomplished using only shifts and adds, with no division required. Consider again n (mod p) where n is 685 and where p is 13. When p is 13, q is equal to 4 and C is equal to 3, from $p = 2^q - C$; $13 = 2^4 - 3 = 16 - 3 = 13$.

The (mod p) arithmetic can be accomplished by using the binary form of n, namely 1010101101. Referring to Figure 7, the shifts and adds are accomplished by first latching the q (4) least significant bits (LSB's) 701 of n, namely 1101. The remaining bits 702 (101010) are multiplied by C (3) to generate product 703 (1111110). Product 703 is added to bits 701 to generate sum 704 (10001011). The q least significant bits 705 (1011) of sum 704 are latched. The remaining bits 706 (1000) are multiplied by C to generate product

707 (11000). Product 707 is added to bits 705 to generate sum 708 (100011). The q least significant bits 709 (0011) of sum 708 are latched. The remaining bits 710 (10) are multiplied by C to generate product 711 (110). Product 711 is added to bits 709 to generate sum 712 (1001). Sum 712, having q or fewer bits, is the solution to $n \pmod p$. $1001 = 2^3 + 2^0 = 8 + 1 = 9$. 685 divided by 13 results in a remainder of 9. The fast class arithmetic provides the solution using only shifts, adds, and multiplies.

Shift and Add Implementation

Fast Mersenne mod operations can be effected via a well known shift procedure. For $p = 2^q - 1$ we can use:

$$x = (x \,\&\, p) + (x >> q) \qquad \text{Equation (17)}$$

a few times in order to reduce a positive x to the appropriate residue value in the interval 0 through $p - 1$ inclusive. This procedure involves shifts and add operations only. Alternatively, we can represent any number $x \pmod p$ by:

$$x = a + b\,2^{(q+1)/2} = (a, b) \qquad \text{Equation (18)}$$

If another integer y be represented as (c, d), we have:

$$xy \pmod{p} = (ac + 2bd, ad + bc) \qquad \text{Equation (19)}$$

after which some trivial shift-add operations may be required to produce the correct reduced residue of xy.

To compute an inverse (mod p), there are at least two ways to proceed. One is to use a binary form of the classical extended-GCD procedure. Another is to use a relational reduction scheme. The relational scheme works as follows:

Given $p = 2^q - 1$, $x \neq 0 \pmod{p}$,

to return $x^{-1} \pmod{p}$:

1) Set $(a, b) = (1, 0)$ and $(y, z) = (x, p)$;

2) If $(y == 0)$ return(z);

3) Find e such that $2^e \,//\, y$;

4) Set $a = 2^{q-e}\, a \pmod{p}$;

5) If$(y == 1)$ return(a);

6) Set $(a, b) = (a+b, a-b)$ and $(y, z) = (y+z, y-z)$;

7) Go to (2).

The binary extended-GCD procedure can be performed without explicit division via the operation $[a/b]_2$, defined as the greatest power of 2 not exceeding a/b :

Given p, and $x \neq 0 \pmod{p}$,

to return $x^{-1} \pmod{p}$:

1) If (x == 1) return(1);

2) Set $(x, v_0) = (0, 1)$ and $(u_1, v_1) = (p, x)$;

3) Set $u_0 = [u_1/v_1]_2$;

4) Set $(x, v_0) = (v_0, x - u_0v_0)$ and $(u_1, v_1) = (v_1, u_1 - u_0v_1)$;

5) If (v_1 == 0) return(x); else go to (3).

The present invention may be implemented on any conventional or general purpose computer system. An example of one embodiment of a computer system for implementing this invention is illustrated in Figure 4. A keyboard 410 and mouse 411 are coupled to a bi-directional system bus 419. The keyboard and mouse are for introducing user input to the computer system and communicating that user input to CPU 413. The computer system of Figure 4 also includes a video memory 414, main memory 415 and mass storage 412, all coupled to bi-directional system bus 419 along with keyboard

410, mouse 411 and CPU 413. The mass storage 412 may include both fixed and removable media, such as magnetic, optical or magnetic optical storage systems or any other available mass storage technology. The mass storage may be shared on a network, or it may be dedicated mass storage. Bus 419 may contain, for example, 32 address lines for addressing video memory 414 or main memory 415. The system bus 419 also includes, for example, a 32-bit data bus for transferring data between and among the components, such as CPU 413, main memory 415, video memory 414 and mass storage 412. Alternatively, multiplex data/address lines may be used instead of separate data and address lines.

In the preferred embodiment of this invention, the CPU 413 is a 32-bit microprocessor manufactured by Motorola, such as the 68030 or 68040. However, any other suitable microprocessor or microcomputer may be utilized. The Motorola microprocessor and its instruction set, bus structure and control lines are described in MC68030 User's Manual, and MC68040 User's Manual, published by Motorola Inc. of Phoenix, Arizona.

Main memory 415 is comprised of dynamic random access memory (DRAM) and in the preferred embodiment of this invention, comprises 8 megabytes of memory. More or less memory may be used without departing from the scope of this invention. Video memory 414 is a dual-ported video

random access memory, and this invention consists, for example, of 256 kbytes of memory. However, more or less video memory may be provided as well.

One port of the video memory 414 is coupled to video multiplexer and shifter 416, which in turn is coupled to video amplifier 417. The video amplifier 417 is used to drive the cathode ray tube (CRT) raster monitor 418. Video multiplexing shifter circuitry 416 and video amplifier 417 are well known in the art and may be implemented by any suitable means. This circuitry converts pixel data stored in video memory 414 to a raster signal suitable for use by monitor 418. Monitor 418 is a type of monitor suitable for displaying graphic images, and in the preferred embodiment of this invention, has a resolution of approximately 1020 x 832. Other resolution monitors may be utilized in this invention.

The computer system described above is for purposes of example only. The present invention may be implemented in any type of computer system or programming or processing environment.

Block Diagram

Figure 8 is a block diagram of the present invention. A sender, represented by the components within dashed line 801, encrypts a plaintext message Ptxt to a ciphertext message C. This message C is sent to a receiver, represented by the components within dashed line 802. The receiver 802 decrypts the ciphertext message C to recover the plaintext message Ptxt.

The sender 801 comprises an encryption/decryption means 803, an elliptic multiplier 805, and a private key source 807. The encryption/decryption means 803 is coupled to the elliptic multiplier 805 through line 809. The elliptic multiplier 805 is coupled to the private key source 807 through line 811.

The encryption/decryption means 804 of receiver 802 is coupled to elliptic multiplier 806 through line 810. The elliptic multiplier 806 is coupled to the private key source 808 through line 812.

The private key source 807 of the sender 801 contains the secure private password of the sender, "*ourPri*". Private key source 807 may be a storage register in a computer system, a password supplied by the sender to the cryptosystem when a message is sent, or even a coded, physical key that is

read by the cryptosystem of Figure 8 when a message is sent or received. Similarly, the private key source 808 of receiver 802 contains the secure private password of the receiver, namely, "*theirPri*".

A separate source 813 stores publicly known information, such as the public keys "ourPub" and "theirPub" of sender 801 and receiver 802, the initial point (X_1, Y_1), the field F_{pk}, and curve parameters a, b, c. This source of information may be a published directory, an on-line source for use by computer systems, or it may be transmitted between sender and receiver over a non-secure transmission medium. The public source 813 is shown symbolically connected to sender 801 through line 815 and to receiver 802 through line 814.

In operation, the sender and receiver generate a shared secret pad for use as an enciphering and deciphering key in a secure transmission. The private key of the sender, *ourPri*, is provided to the elliptic multiplier 805, along with the sender's public key, theirPub. The elliptic multiplier 805 computes an enciphering key e_K from (*ourPri*) ° (theirPub) (mod p). The enciphering key is provided to the encryption/decryption means 803, along with the plaintext message Ptxt. The enciphering key is used with an encrypting scheme, such as the DES scheme or the elliptic curve scheme of

the present invention, to generate a ciphertext message C. The ciphertext message is transmitted to the receiver 802 over a nonsecure channel 816.

The receiver 802 generates a deciphering key D_K using the receiver's private key, *theirPri*. *TheirPri* is provided from the private key source 808 to the elliptic multiplier 804, along with sender's public key, **ourPub**, (from the public source 813). Deciphering key D_K is generated from (*theirPri*) ° (**ourPub**) (mod p). The deciphering key D_K is equal to the enciphering key e_K due to the abelian nature of the elliptic multiplication function. Therefore, the receiver 802 reverses the encryption scheme, using the deciphering key D_K, to recover the plaintext message Ptxt from the ciphertext message C.

The encryption/decryption means and elliptic multiplier of the sender 801 and receiver 802 can be implemented as program steps to be executed on a microprocessor.

Inversionless Parameterization

The use of fast class numbers eliminates division operations in (mod p) arithmetic operations. However, as illustrated by equations 13-16 above, the elliptic multiply operation "°" requires a number of division operations to be performed. The present invention reduces the number of divisions

required for elliptic multiply operations by selecting the initial parameterization to be inversionless. This is accomplished by selecting the initial point so that the "Y" terms are not needed.

In the present invention, both sender and recipient generate a shared secret pad, as a particular x-coordinate on the elliptic curve. By choosing the initial point (X_1, Y_1) appropriately, divisions in the process of establishing multiples $n \circ (X_1, Y_1)$ are eliminated. In the steps that follow, the form

$$n \circ (X_m/Z_m) \qquad \text{Equation (20)}$$

for integers n, denotes the coordinate (X_{n+m}/Z_{n+m}). For $x = X/Z$ the x-coordinate of the multiple $n(x, y)$ as X_n/Z_n , is calculated using a "binary ladder" method in accordance with the adding-doubling rules, which involve multiply mod operations. For the Montgomery parameterization (Equation 7c), these rules are:

$$\text{If } i \neq j: \quad X_{i+j} = Z_{i-j}\,(X_iX_j - Z_iZ_j)^2 \qquad \text{Equation (21)}$$

$$Z_{i+j} = X_{i-j}\,(X_iZ_j - Z_iX_j)^2 \qquad \text{Equation (22)}$$

Otherwise, if $i = j$:

$$X_{2i} = (X_i^2 - Z_i^2)^2 \qquad \text{Equation (23)}$$

$$Z_{2i} = 4\, X_i Z_i\, (X_i^2 + c\, X_i Z_i + Z_i^2) \qquad \text{Equation (24)}$$

These equations do not require divisions, simplifying the calculations when the present invention is implemented in the present preferred embodiment. This is known as "inversionless parameterization" (due to the absence of division operations), and is described in *"Speeding the Pollard and Elliptic Curve Methods of Factorization"* Montgomery, P. 1987 *Math. Comp.*, 48 (243-264). When the field is simply F_p this scheme enables us to compute multiples nx via multiplication, addition, and (rapid) Mersenne mod operations. This also holds when the field is $F_p 2$. Because $p = 3 \pmod 4$ for any Mersenne prime p, we may represent any X_i or Z_i as a complex integer, proceeding with complex arithmetic for which both real and imaginary post-multiply components can be reduced rapidly (mod p). We also choose $Z_1 = 1$, so that the initial point on the curve is $(X_1/1, y)$ where y will not be needed.

Using both fast class numbers and inversionless parameterization, a public key exchange using the method of the present invention can proceed as follows. In the following example, the prime is a Mersenne prime. However, any of the fast class numbers described herein may be substituted.

1) At "our" end, use curve parameters (a, b, c), to compute a public key: ourPub $\varepsilon F_{pk} \times F_{pk}$

$(X/Z) = ourPri \circ (X_1/1)$

ourPub $= XZ^{-1}$

2) At "their" end, use parameters (a, b, c) to compute a public key: theirPub $\varepsilon F_{pk} \times F_{pk}$

$(X/Z) = theirPri \circ (X_1/1)$

theirPub $= XZ^{-1}$

3) The two public keys ourPub and theirPub are published, and therefore are known.

4) Compute a shared secret pad: ourPad $\varepsilon F_{pk} \times F_{pk}$

$(X/Z) = ourPri \circ (\text{theirPub}/1)$

ourPad $= XZ^{-1}$

5) Compute a shared secret pad: theirPad $\varepsilon F_{pk} \times F_{pk}$

$(X/Z) = theirPri \circ (ourPub/1)$

$theirPad = XZ^{-1}$

The usual key exchange has been completed, with

$ourPad = theirPad$

Message encryption/decryption between "our" end and "their" end may proceed according to this mutual pad.

FFT Multiply

For very large exponents, such as $q > 5000$, it is advantageous to perform multiplication by taking Fourier transforms of streams of digits. FFT multiply works accurately, for example on a 68040-based NeXTstation, for general operations $xy \pmod p$ where $p = 2^q - 1$ has no more than $q = 2^{20}$ (about one million) bits. Furthermore, for Mersenne p there are further savings when one observes that order-q cyclic convolution of binary bits is equivalent to multiplication (mod 2^q-1). The use of FFT multiply techniques results in the ability to perform multiply-mod in a time roughly proportional to $q \log q$, rather than q^2.

Elliptic curve algebra can be sped up *intrinsically* with FFT techniques. Let $\underline{X}$ denote generally the Fourier transform of the digits of X, this transform being the same one used in FFT multiplication. Then we can compute coordinates from equations 21-24. To compute X_{i+j} for example, we can use five appropriate transforms, (X_i, X_j, Z_i, Z_j, and Z_{i-j}) (some of which can have been stored previously) to create the transform:

$$\underline{X}_{i+j} = Z_{i-j}\,(\underline{X}_i\underline{X}_j - \underline{Z}_i\underline{Z}_j)^2$$

In this way the answer X_{i+j} can be obtained via 7 FFT's. (Note that the usual practice of using 2 FFT's for squaring and 3 FFT's for multiplication results in 11 FFT's for the "standard" FFT approach). The ratio 7/11 indicates a significant savings for the intrinsic method. In certain cases, such as when p is a Mersenne prime and one also has an errorless number-theoretic transform available , one can save spectra from the past and stay in spectral space for the duration of long calculations; in this way reducing times even further.

A flow diagram illustrating the operation of the present invention when using fast class numbers, inversionless parameterization and FFT multiply operations is illustrated in Figure 9. At step 901, a fast class number p is chosen where $p = 2^q - C$. The term q is the bit depth of the encryption

scheme. The greater the number of bits, the greater the security. For large values of q, FFT multiply operations are used to calculate p. The term p is made publicly available.

At step 902, the element k for the field F_{p^k} is chosen and made public. At step 903, an initial point (X_1/Z) on the elliptic curve is selected. By selecting the initial point to be inversionless, costly divides are avoided. The initial point is made public. The curve parameter a is chosen at step 904 and made public.

At step 905, the sender computes $X_1/Z = ourPri \circ (X_1/1)$ using inversionless parameterization. The sender's public key is generated **ourPub** $= (XZ^{-1})(\text{mod } p)$. The receiver's public key **theirPub** $= (XZ^{-1})(\text{mod } p)$, is generated at step 906.

A one time pad for the sender, **ourPad**, is generated at step 907. $X/Z = (ourPri) \circ (\textbf{theirPub}/1)$. **ourPad** $= XZ^{-1}(\text{mod } p)$. At step 908, a one time pad for the receiver, **theirPad**, is generated. $X/Z = (theirPri) \circ (\textbf{ourPub}/1)$. **theirPad** $= XZ^{-1}(\text{mod } p)$. The calculation of **ourPad** and **theirPad** utilizes FFT multiplies to eliminate the need to calculate the inversion Z^{-1}. At step 909, the sender converts a plaintext message Ptxt to a ciphertext message C using **ourPad**. The ciphertext message C is transmitted to the receiver. At step 910, the receiver

recovers the plaintext message Ptxt by deciphering the ciphertext message C using theirPad.

FEE Security

The algebraic factor $M_{89} = 2^{89}-1$, which is a Mersenne prime, occurs with "natural" statistics when the elliptic curve method (ECM) was employed. This was shown in attempts to complete the factorization of $M_{445} = 2^{445} - 1$. In other words, for random parameters c (using Montgomery parameterization with $a = 0, b = 1$) the occurrence $k(X_1/1) = O$ for elliptic curves over F_p with $p = M_{89}$ was statistically consistent with the asymptotic estimate that the time to find the factor M_{89} of M_{445} be $O(\exp(\sqrt{(2 \log p \log \log p)})$. These observations in turn suggested that finding the group order over F_p is not "accidentally" easier for Mersenne primes p, given the assumption of random c parameters.

Secondly, to check that the discrete logarithm problem attendant to FEE is not accidentally trivial, it can be verified, for particular c parameters, that for some bounded set of integers N

$$(p^N - 1)(X_1/1) \neq O$$

The inequality avoids the trivial reduction of the discrete logarithm evaluation to the equivalent evaluation over a corresponding finite field. Failures of the inequality are extremely rare, in fact no non-trivial instances are known at this time for $q > 89$.

The present invention provides a number of advantages over prior art schemes, particularly factoring schemes such as the RSA scheme. The present invention can provide the same security with fewer bits, increasing speed of operation. Alternatively, for the same number of bits, the system of the present invention provides greater security.

Another advantage of the present cryptosystem over prior art cryptosystems is the distribution of private keys. In prior art schemes such as RSA, large prime numbers must be generated to create private keys. The present invention does not require that the private key be a prime number. Therefore, users can generate their own private keys, so long as a public key is generated and published using correct and publicly known parameters. A user cannot generate its own private key in the RSA system.

DIGITAL SIGNATURE

The present invention provides an improved method for creating and authenticating a digital signature that uses the elliptic algebra described above and a hashing or digesting function. The sender has prepared an encrypted message "ciphertext". This message may be encrypted as described above or may be encrypted using any other encryption scheme. The sender then creates a digital signature to append to the message as a way of "signing" the message. The signature scheme of the preferred embodiment is described below, followed by the method of reducing computations.

Creation of Signature

Assume a curve parameterized by a, b, c with starting point $(X_1/1)$. Also assume the starting point to have order N on the elliptic curve. The sender's public key **ourPub** is generated as the multiple $ourPri \circ (X_1/1)$, where *ourPri* is our private key (an integer) and $\circ$ is multiplication on the elliptic curve. The digital signature is created as follows:

1) Choose a random integer m of approximately q bits.

2) Compute the point

$$P = m \circ (X_1/1).$$

3) Using a message digest function M, compute the integer

$$u = (m + our\ Pri * M(\text{ciphertext}, P)) \pmod{N}$$

where ciphertext is the encrypted message to be sent.

4) Along with the ciphertext, transmit the digital signature as the pair (u, P). Note that u is an integer of about 2^q bits, while P is a point on the curve.

In the preferred embodiment of the present invention, a message digesting function M such as MD2 or MD5 is used as part of the creation of the digital signature. However, the present invention may be implemented using other digesting functions or by using any suitable hashing function.

Authentication of Digital Signature

The receiver attempts to authenticate the signature by generating a pair of points to match the digital signature pair, using the ciphertext message and

the public key of the purported sender. The receiver verifies the signature using the following steps:

1) Using the u part of the signature, compute the point

$$Q = u \circ (X_1/1)$$

2) Compare the point Q to the point

$$R = P + M(\text{ciphertext}, P) \circ \mathbf{ourPub}$$

The signature is invalid if these elliptic points Q and R do not compare exactly. In other words, if the signature is authentic, the following must hold:

$$u \circ (X_1/1) = P + M(\text{ciphertext}, P) \circ \mathbf{ourPub}$$

Substituting for u on the left side of the equation above gives:

$$(m + our\ Pri * M(\text{ciphertext}, P)) \circ (X_1/1) = P + M(\text{ciphertext}, P) \circ \mathbf{ourPub}$$

or:

$m \circ (X_1/1) + (ourPri*M(\text{ciphertext, P})) \circ (X_1/1) = P + M(\text{ciphertext, P}) \circ$ **ourPub**

Substituting for **ourPub** on the right side of the equation yields:

$m \circ (X_1/1) + (ourPri*M(\text{ciphertext, P})) \circ (X_1/1) = P + M(\text{ciphertext, P}) \circ ourPri \circ (X_1/1)$

Since $P = m \circ (X_1/1)$ from above, the left side becomes:

$P + (ourPri*M(\text{ciphertext, P})) \circ (X_1/1) = P + M(\text{ciphertext, P}) \circ ourPri \circ (X_1/1)$

Moving *ourPri* in the right side of the equation gives:

$P + ourPri*M(\text{ciphertext, P})) \circ (X_1/1) = P + ourPri*M(\text{ciphertext, P}) \circ (X_1/1)$

Thus, a point on a curve is calculated via two different equations using the transmitted pair (u, P). It can be seen that by calculating Q from the transmitted point u, and by calculating R from transmitted point P, the

ciphertext message, and the public key of the purported sender, the digital signature is assumed authenticated when Q and R match.

Security

The digital signature scheme of this scheme is secure on the basis of the following observation. To forge a signature one would need to find a pair (u, P) and a ciphertext that satisfy the equation

$$u \circ (X_1/1) = P + M(\text{ciphertext}, P) \circ \text{ourPub}$$

This would either entail an elliptic logarithm operation (the basis of the encryption security of the present invention) or breaking of the hash function M.

Optimizing Authentication

The recipient's final step in the digital signature scheme of the present invention involves the addition of two points; namely P and M(ciphertext, P) ° ourPub to yield R and comparing that sum to a point Q. One could perform the elliptic addition using specified y-coordinates at each step. The scheme of the present invention provides a method of deducing the possible values of

the x-coordinate of a sum of two points, using only the respective x-coordinates of the original two points in question. Using this method one may rapidly perform a necessity check on whether the points Q and the sum of P + M(ciphertext, P) ° ourPub have identical x-coordinates.

A principle for fast verification of sums, using only x-coordinates, runs as follows. For example, using Montgomery parameterization, let the curve be

$$y^2 = x^3 + cx^2 + x$$

Theorem: Let $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$, and $Q = (x, y)$ be three points on a given curve, with $x_1 \neq x_2$, Then

$$P_1 + P_2 = Q$$

only if

$$x\ (C - x) = B^2$$

where

$$B = (x_1x_2 - 1) / (x_1 - x_2)$$

$$C = 2((x_1x_2 + 1)(x_1 + x_2 + 2c) - 2c) / (x_1 - x_2)^2$$

The proof is given as follows. Not knowing the y-coordinates of P_1 and P_2, the only possibilities for the x-coordinate of the sum $P_1 + P_2$ are, for any fixed pair (y_1, y_2), the respective x-coordinates (call them e,f) of the two forms $(x_1,y_1) \pm (x_2, y_2)$. One can compute:

$$ef = B^2$$

$$e + f = C$$

as in Montgomery, *supra*. Since x is one or the other of e, f it is necessary that $(x - e)(x - f) = 0$, whence the quadratic equation of the theorem holds.

Thus, when using the x-coordinate scheme of the present invention, it is possible to have two solutions that satisfy $(x - e)(x - f) = 0$. One possible solution is therefore generated from an inauthentic signature. However, because there are literally millions of possible solutions, when $(x - e)(x - f) = 0$ is satisfied, it can be safely assumed that the signature is authentic.

In practical application, P_1 represents the calculated point P that is sent as part of the signature by the sender. P_2 represents the expression

M(ciphertext, P) ° ourPub. Q of course represents u ° $(X_1/1)$. $P_1 + P_2$ represents R and is compared to Q.

Flow Diagrams

Figure 10 is a flow diagram illustrating the generation of a digital signature using the present invention. At step 1001, the sender chooses a random integer m. This random integer can be generated using a suitable random number generator for use with a microprocessor. At step 1002 a point P is calculated using m. As noted above, this point is generated using the relation $P = m \circ (X_1/1)$. in the preferred embodiment of the present invention. However, other schemes may be used for generating point P without departing from the scope of the present invention.

At step 1003, a second number, u, is calculated using m, P, *ourPri*, and the ciphertext message. In the preferred embodiment of the invention, this is generated using the relationship $u = m + our\ Pri * M$(ciphertext, P). As noted above, hashing functions other than digesting functions MD2 and MD5 can be used. In addition, other relationships can be used to calculate u. It is recommended that if other relationships are used, that m, P, *ourPri* and the ciphertext message be used. At step 1004, the calculated pair (u, P) is sent as a digital signature.

Figure 11 is a flow diagram illustrating the authentication of a digital signature in the present invention. At step 1101 the recipient of the message receives the digital signature (u, P) and the ciphertext message. At step 1102 the point Q is generated using the point u. In the preferred embodiment, the relationship $Q = u \circ (X_1/1)$ is used to generate Q. Other relationships may be used depending on what relationships were used to calculate u, P by the sender.

At step 1103, a point P_2 is generated using **ourPub** and the ciphertext message. In the preferred embodiment, the relationship M(ciphertext, P) ° **ourPub** is used to generate P_2. Other relationships may be used depending on what relationships were used to calculate u, P by the sender.

At step 1104, the x values of P_1 and P_2 are used to determine values B and C and ultimately, e and f. This leads to two possible x values for the sum of P_1 and P_2. At decision block 1105 the argument "$e,f = x$?" is made to determine if either of the possible x values satisfies the equality of $P_1 + P_2 = Q$. If neither of the calculated x values satisfy the equation, that is, if the argument at decision block 1105 is false, the signature is not authentic and is indicated at block 1106. If one of the x values does satisfy the equation, that is,

if the argument at decision block 1105 is true, a valid signature is assumed and indicated at block 1107.

Block Diagram

Figure 12 illustrates a block diagram for implementing the digital signature scheme of the present invention. Where elements of Figure 12 are in common with elements of Figure 8, the same element numbers are used. The signature scheme is shown in use with an encryption scheme that uses elliptic multiplication, but this is by way of example only. The present invention can be used with any type of encryption scheme.

A sender, represented by the components within dashed line 1201, encrypts a plaintext message Ptxt to a ciphertext message C and generates a signature (u, P). This message C and signature (u, P) is sent to a receiver, represented by the components within dashed line 1202. The receiver 1202 decrypts the ciphertext message C to recover the plaintext message, and authenticates the signature (u, P).

The sender 1201 comprises an encryption/decryption means 1203, an elliptic multiplier 805, a random number generator 1205, a hasher 1207, and a private key source 807. The encryption/decryption means 1203 is coupled to

the elliptic multiplier 805 through line 809. The elliptic multiplier 805 is coupled to the private key source 807 through line 811. The random number generator 1205 provides random number *m* on line 1209 to elliptic multiplier 805 and to hasher 1207. Elliptic multiplier 805 provides the number *u* to the nonsecure channel 816 via line 1211. The encrypted ciphertext C is provided to hasher 1207 via line 1213. Hasher 1207 provides point P to nonsecure channel 816 via line 1215.

The encryption/decryption means 1204 of receiver 1202 is coupled to elliptic multiplier 806 through line 810. The elliptic multiplier 806 is coupled to the private key source 808 through line 812. The number *u* is provided to the elliptic multiplier 806 from the nonsecure channel 816 via line 1212. Elliptic multiplier 806 generates point Q and provides it to comparator 1208 via line 1216. Hasher 1206 receives the ciphertext message C and point P from nonsecure channel 816 via line 1210, and ourPub from source 813 via line 1218. Hasher 1206 outputs point R to comparator 1208 via line 1214.

The private key source 807 of the sender 801 contains the secure private password of the sender, *"ourPri"*. Private key source 807 may be a storage register in a computer system, a password supplied by the sender to the cryptosystem when a message is sent, or even a coded, physical key that is read by the cryptosystem of Figure 12 when a message is sent or received.

Similarly, the private key source 808 of receiver 802 contains the secure private password of the receiver, namely, "*theirPri*".

A separate source 813 stores publicly known information, such as the public keys "**ourPub**" and "**theirPub**" of sender 1201 and receiver 1202, the initial point (x_1, y_1), the field F_{p^k}, and curve parameters a, b, c. This source of information may be a published directory, an on-line source for use by computer systems, or it may transmitted between sender and receiver over a non-secure transmission medium. The public source 813 is shown symbolically connected to sender 1201 through line 815 and to receiver 1202 and hasher 1206 through lines 814 and 1218 respectively.

In operation, the sender and receiver generate a common one time pad for use as an enciphering and deciphering key in a secure transmission, as described above. The enciphering key is provided to the encryption/decryption means 1203, along with the plaintext message. The enciphering key is used with an encrypting scheme, such as the DES scheme or the elliptic curve scheme of the present invention, to generate a ciphertext message C. The random number generator 1205 generates random number m and provides it to elliptic multiplier 805. Elliptic multiplier 805 generates number u and provides it to the receiver via nonsecure channel 816. The ciphertext message C is provided to the hasher 1207, along with the random

number *m* and *ourPri*. Hasher 1207 generates point P and provides it to nonsecure channel 816. The ciphertext message, along with signature (u, P), is transmitted to the receiver 1202 over a nonsecure channel 816.

The receiver 1202 generates a deciphering key D_K using the receiver's private key, *theirPri*. *TheirPri* is provided from the private key source 808 to the elliptic multiplier 806, along with sender's public key, ourPub, (from the public source 813). Deciphering key D_K is generated from (*theirPri*) ° (ourPub) (mod p). The deciphering key D_K is equal to the enciphering key e_K due to the abelian nature of the elliptic multiplication function. Therefore, the receiver 1202 reverses the encryption scheme, using the deciphering key D_K, to recover the plaintext message from the ciphertext message C.

The elliptic multiplier 806 of the receiver 1202 receives the number u from the nonsecure channel 816. The elliptic multiplier 806 generates point Q and provides it to comparator 1208. Hasher receives the ciphertext message C and point P from the nonsecure channel 816 and the purported senders public key ourPub from source 813 and generates point R, which it provides to comparator 1208. Comparator 1208 compares points Q and R and if they match, the signature is assumed to be valid. In the present invention, the comparison of points Q and R is accomplished using the optimized scheme using x values described above.

The encryption/decryption means and elliptic multiplier of the sender 1201 and receiver 1202 can be implemented as program steps to be executed on a microprocessor.

DIRECT EMBEDDING

The present invention takes advantage of the fact that parcels of text can be mapped to one of two curves $E^{\pm}$. Using a receiver's public key, the sender generates and sends as a triple a message coordinate, a clue value, and a sign to the receiver. Using the clue and the receiver's private key, the text parcel may be decrypted from the message coordinate. In the expansionless form, the sender and receiver use their shared secret pad to compute shared clues so that each message coordinate is sent with a one-bit sign.

Elliptic curves generated using fast elliptic algebra described above are sufficiently special (i.e. tightly defined), that any parcel of plaintext will embed directly and naturally on one of only two possible curves. We call these curves the "+" curve and the "-" curve.

Expansionless Direct Embedding

A further refinement of this invention reduces the size of the encrypted parcels by eliminating the clue component of each triple. This is achieved by establishing means of generating clues in synchrony between sender and receiver. For example, we may use the method of direct embedding above to securely send two random numbers r, s from the sender to the receiver. The sender computes the first clue as

$$\text{clue}_1 = r \circ ourPri \circ \text{theirPub}$$

and similarly, the receiver computes the same clue as

$$\text{clue}_1 = r \circ theirPri \circ \text{ourPub}$$

Subsequent clues can be formed on both sides by performing an elliptic addition operation as follows:

$$\text{clue}_{n+1} = \text{clue}_n + s \circ P$$

where P is the initial point on the curve.

As in direct embedding, an embedded message parcel point P_{text} is ellipticly added to clue to form the message point and the recovery clue g is computed and sent, now as only a pair: (message, g).

Elliptic Curve Operations

As is shown below, the sender of direct embedded plaintext sends a bit to identify which of the curves is to be used at the receiver's end so that decryption can be accomplished.

The idea of direct embedding is to embed parcels of plaintext on the elliptic curve itself. Say that a point P_{text} is a curve point that contains a parcel of plaintext to be encrypted. Using fast elliptic curve algebra described above, it is possible to transmit the triple consisting of a pair of points and a single bit g:

$$(P_{text} + r^{\circ}\ \text{theirPub}, r^{\circ}\ P_1, g)$$

(in practice, only sending pairs of x-coordinates and one bit are sent rather than pairs of curve points *per se* and one bit) where r is a random integer. Think of this transmission qualitatively as:

(message, clue, parity)

At "their" end, the receiver uses the "clue" and "parity" and their private key "*theirPri*" to deduce the plaintext from the "message." The parity bit is an overall sign result depending on the curve ambiguity and the sign ambiguities of quadratic forms.

The direct embedding of text is understood by reviewing the following elliptic curve relationships.

We concentrate herein on the case of fields Fp, where

$$p = 2^q - 1$$

is the Mersenne prime. The elliptic curve in question, call it E, is assembled to be comprised of points $P = (x,y) \varepsilon F_p \times F_p$ satisfying:

$$s\, y^2 = x^3 + c\, x^2 + b\, x + a$$

where the sign s of the curve is restricted to be either +1 or -1, and $c \neq 2$, together with a "point at infinity" O. Note that we use boldface for actual curve points. That is, the notation so far means:

x, y are both integers (mod p)

P, a point on the curve, is a pair (x,y),

or possibly the abstract "point at infinity" O

E is the set of all P

A powerful classical result is Hasse's theorem, that $|E|$ the order of the curve; i.e., the total number of points P, is close to p itself in the sense that

$$\left| |E| - p - 1 \right| \leq 2\sqrt{p}$$

The elliptic curve E, if equipped with a certain operation between its points, becomes an additive group. The point O is the additive identity, while the group law of addition works as follows. For two non-O points

$$P_1 = (x_1, y_1) \qquad P_2 = (x_2, y_2)$$

we define the curve addition

$$P_3 = P_1 + P_2 = (x_3, y_3)$$

and subtraction

$$P_4 = P_1 - P_2 = (x_4, y_4)$$

via the relations for the Montgomery parameterization $a = 1$, $b = 0$:

$$x_3 = s\left(\frac{y_1 - y_2}{x_1 - x_2}\right)^2 - c - x_1 - x_2; \text{ if } x_1 \neq x_2$$

$$x_3 = \frac{\left(x_1^2 - 1\right)^2}{4x_1\left(x_1^2 + cx_1 + 1\right)}; \text{ if } x_1 = x_2$$

together with the negation rule:

$$-P_1 = (x_1, -y_1)$$

It may then be derived that, for $x_1 \neq x_2$ the sum and difference x-coordinates are related via:

$$x_3 x_4 = F(x_1, x_2) := \left(\frac{x_1 x_2 - 1}{x_1 - x_2}\right)^2$$

$$x_3 + x_4 = G(x_1, x_2) := \frac{2}{(x_1 - x_2)^2} \; ((x_1x_2 + 1)(x_1 + x_2 + 2c) - 2c)$$

These defined functions F, G figure in the theory of direct embedding. Note that the G function in particular can be written alternatively, as:

$$G(x_1, x_2) = \frac{2}{(x_1 - x_2)^2} \; (x_1 Q(x_2) + x_2 Q(x_1))$$

where the Q function is the defining quadratic form for the elliptic curve; viz.

$$Q(z) = z^2 + cz + 1$$

Elliptic multiplication ladder

In actual implementations, rapid elliptic curve multiplication is performed via the inversionless parameterization system of [Montgomery 1987], in which the y-coordinate is ignored. For some point $P = (X_1, y)$ we define the n-th multiple of P , denoted $n \circ P$, as the elliptic sum of n copies of P. (When integer $n = 0$ we interpret $0 \circ P$ as the abstract point O). Now denote the x-coordinate of the multiple $n \circ P$ by X_n/Z_n with $Z_1 = 1$ understood. The integers X_n, Z_n can be evaluated via a binary ladder method in accordance with certain adding-doubling rules. These rules can be derived

from the basic addition/subtraction laws previous, and take the form of equations 21, 22, 23 and 24 above.

Mersenne mod operation

The elliptic multiplication ladder involves, through the adding-doubling rules, multiply-mod operations. These operations can be made efficient. Also the single inverse Z^{-1} required to resolve a key or pad can be effected rapidly. In any case, all arithmetic may proceed with multiplications, shifts, and adds; resulting in a division-free system.

Fast Mersenne mod operations can be effected as described in connection with equations 17, 18, and 19 above. An inverse (mod p) can be computed as described above following equation 19.

There is also a recursive-inverse algorithm, based on polynomial-GCD methods, which in actual practice takes time $O(q\log^m q)$ for some small integer m. The inverse times are competitive with cumulative FFT multiply techniques such as described above.

THEOREMS

The following theorems provide support for the direct embedding scheme of the present invention.

Theorem 1

For a point P on elliptic curve E, and integers m, n we have

$$mn \circ P = nm \circ P = m(n \circ P) = n(m \circ P)$$

This illustrates the rules of commutativity and associativity.

Theorem 2

For given parameters $a = 0$, $b = 1$, $c \neq 2$, an arbitrary integer x is a valid x -coordinate of some point lying on one of the two fast elliptic curves:

$$E^{\pm}: \pm y^2 = xQ(x)$$

Note that because p is a Mersenne prime and thus = 3(mod 4), an integer $s = xQ(x)$ is either a square or its negative is.

Theorem 3

Let $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ and assume $P_3 = P_1 + P_2 = (x_3, y_3)$. Then x_3 must satisfy:

Case $x_1 \neq x_2$: The quadratic relation $x_3(G(x_1, x_2) - x_3) = F(x_1, x_2)^2$

Case $x_1 = x_2$: The relation $x_3 = (x_1^2 - 1)^2/(4x_1Q(x_1))$

Theorem 4

Let $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ and $x_1 \neq x_2$. Denote:

$P_3 = P_1 + P_2 = (x_3, y_3)$.

$P_4 = P_1 - P_2 = (x_4, y_4)$.

Then x_3 is one of the following two values:

$$(x_1Q(x_2) + x_2Q(x_1) \pm 2(x_1Q(x_1)x_2Q(x_2))^{(p+1)/4})/((x_1 - x_2)^2)$$

while x_4 is the other value.

The above results from solving the quadratic relation of Theorem 3 above to give:

$$x_3 = \frac{F \pm \sqrt{F^2 - 4G^2}}{2}$$

The square root of a square $a \pmod p$, for $p = 3 \pmod 4$ is always plus or minus $a^{(p+1)/4} \pmod p$.

Theorem 5

For arbitrary integer $x \pmod p$, being on one of the curves $E^{\pm}$, the correct sign + or - is given by whichever sign holds in:

$$(xQ(x))^{(p+1)/2} = \pm xQ(x) \pmod p$$

If $xQ(x)$ is 0, the + curve is used.

DIRECT EMBEDDING ALGORITHM

Assume:

Parameters q, a=0, b:=±1, $c \neq 2$, giving rise to two possible elliptic curves $E^{\pm}$;

Two public initial x-coordinates X1± such that two points

$$P_1^+ = (X_1^+, ?) \qquad P_1^- = (X_1^- ?)$$

lie respectively on the curves $E^{\pm}$. Note that the y coordinate (denoted as "?") can be ignored when using fast elliptic algebra.

The existence of a pair of public keys: theirPub± generated from the single private key *theirPri* according to:

$$(\text{theirPub}^{\pm}, ?) = \textit{theirPri} \cdot P_1^{\pm}$$

Plaintext is to be broken up into parcels of no more than q bits each; i.e., a parcel is a value (mod p).

The existence of a function elliptic_add (x_1, x_2, s) that computes one of x_3 or x_4 from Theorem 4 above by using the sign s; or from Theorem 3 in the rare case $x_1 = x_2$.

To encrypt a plaintext parcel x_{text}:

1. Determine, from Theorem 5, for which of the two curves $E^{\pm}$ the parcel x_{text} is a valid x-coordinate. Denote said curve by sign $s = \pm 1$.

2. Choose random r, and calculate $(x_q, ?) := r \cdot (\text{theirPub}^{\pm}, ?)$, using the ladder arithmetic described above.

3. Calculate a message coordinate $x_m := \text{elliptic\_add}(x_{text}, x_q, +1)$

4. Calculate a clue x_c using the random r and the public points as $(x_c, ?) := r \cdot P_1^{\pm}$.

5. Determine which sign in elliptic_add $(x_m, x_q, \pm 1) = x_{text}$ is valid and call this sign g.

6. Transmit the triple of the message coordinate, the clue and the sign to the sender as (x_m, x_c, g)

To decrypt a parcel of plaintext:

1. Assuming receipt of (x_m, x_c, g), use the clue x_c to compute an x-coordinate x_q from

$$(x_q, ?) := theirPri \cdot (x_c, ?)$$

2. Recover plaintext as:

$$x_{text} := \text{elliptic\_add}\ (x_m, x_c, g)$$

A software implementation of Algorithm 3 is attached at the end of this document, as Appendix code. In actual implementation, it is possible to perform the random integer elliptic multiplication (by r) only periodically, so as to reduce execution time. It turns out in practice that the multiplication by r is actually more costly even than the exponentiation embodied in, say, Theorem 4.

Flow Diagram For Direct Embedding

Figure 13 is a flow diagram of encrypting a plaintext message using direct embedding. At step 1301 divide the plaintext into parcels of no more

than q bits each. For each parcel x_{text}, determine at step 1302 for which of two curves $E\pm$ the parcel is a valid x -coordinate. This can be accomplished by use of Theorem 5 described above. At step 1303 denote the appropriate curve by sign = ±1.

At step 1304 choose a random r. At step 1305 calculate $(x_q, ?) = r \circ$ (**theirPub**±, ?). This may be accomplished using ladder arithmetic. At step 1306 calculate a message coordinate x_m. This can be accomplished by $x_m :=$ elliptic_add(x_{text} , x_q, +1) or any other suitable method. The clue is calculated at step 1307. This may accomplished by $(x_c, ?) := r \circ P_1^{\pm}$.

At step 1308 define as g the sign that holds for x_{text}. This sign may be determined by testing if elliptic_add (x_m, x_q, +1) equals x_{text}. At step 1309 transmit the message coordinate, the clue, and the sign as a triple to the receiver in the form (x_m, x_c, g).

Figure 14 is a flow diagram of decrypting the encrypted message of Figure 13. At step 1401 receive a triple (x_m, x_c, g) from sender. At step 1402 compute an x-coordinate x_q from $(x_q, ?) :=$*theirPri* $\circ$ $(x_c, ?)$. The plaintext can then be recovered at step 1403 by x_{text} := elliptic_add (x_m, x_c, g).

Expansionless Direct Embedding Algorithm

To encrypt a series of plaintext parcel x_{text_n}:

1. Select two random numbers $r, s \in F_{pk}$ and transmit them to the receiver, for instance, by using the direct embedding algorithm.

2. Compute initial clues for both curves by computing

$$clue = r \circ ourPri \circ theirPub^{\pm}$$

$$s = s \circ P^{\pm}$$

3. For plaintext parcel x_{text_i} determine for which of the two curves x_{text_i} is a valid point (or, for which x_{text_i} is a valid coordinate).

4. Using the correct curve points, calculate a message coordinate

$$m_i := \text{elliptic\_add}(x_{text_i}, clue_i, +1)$$

5. Determine which sign in elliptic_add(m_i, $clue_i$, ±1) recovers x_{text_i} and call this sign g.

6. Transmit the pair (m_i, g).

7. For subsequent parcels, compute

$$clue_{i+1} = \text{elliptic\_add}(r \circ clue_i, s, +1)$$

and repeat steps 3-6.

To decrypt a series of pairs (m, g):

1. Recover random numbers r, s and compute initial clues as follows:

$$clue = r \circ theirPri \circ ourPub^{\pm}$$

$$s = s \circ P^{\pm}$$

2. For each pair (m, g), determine which curve holds the point m.

3. Recover plaintext via the following operation upon the points from the determined curve:

$$\text{elliptic\_add}(clue, m, g)$$

4. Recompute clue for the subsequent pair

$$clue_{i+1} = \text{elliptic\_add}(r \circ clue_i, s, +1)$$

Flow Diagram For Expansionless Direct Embedding

Figure 15 is a flow diagram of encrypting a plaintext message using expansionless direct embedding. At step 1500, the plaintext is divided into parcels of no more than q bits each. Two random numbers, r and s, are selected at step 1501, and transmitted to the receiver in step 1502. At step 1503, initial clues, clue and s, are determined for both curves using random numbers r and s.

At step 1504, for each parcel x_{text}, the sender determines for which of two curves $E\pm$ the parcel is a valid x-coordinate. The message coordinate is calculated at step 1505 using $m_i := \text{elliptic\_add}(x_{text_i}, clue_i, +1)$. At step 1506, the sender determines which sign in $\text{elliptic\_add}(m_i, clue_i, \pm 1)$ recovers x_{text_i} and denotes the sign as g.

At step 1507, the current pair (m, g) is transmitted to the receiver, and the next clues are computed in step 1508 using the previous clue and random

numbers *r* and *s*. After step 1508, the process returns to step 1504 to encrypt the next parcel.

Figure 16 is a flow diagram for decrypting the encrypted message of Figure 15. At step 1600, the receiver recovers the random numbers *r* and *s* transmitted by the receiver, and at step 1601, random numbers *r* and *s* are used to determine the initial clues. In step 1602, the receiver recovers the pair (m, *g*) transmitted by the sender. For the current pair (m, *g*), at step 1603, the receiver determines which of the elliptic curves holds the point m. Using points from the determined curve, the plaintext parcel is recovered at step 1604 using elliptic_add(clue, m, *g*). In step 1605, the next clues are computed based on the previous clues and random numbers *r* and *s*. After step 1605, the process returns to step 1602 for the next parcel.

Code

An example of code written in Mathematica for implementing encryption and decryption using direct embedding is as follows:

```
(* Elliptic algebra functions: FEED format.

y^2 = x^3 + c x^2 + a x + b.

Montgomery: b = 0, a = 1;
Weierstrass: c = 0;
```

```
  Atkin3: c = a = 0;
  Atkin4: c = b = 0;

  Parameters c, a, b, p must be global.
*)

elleven[pt_] := Block[{x1 = pt[[1]], z1 = pt[[2]], e, f },
        e = Mod[(x1^2 - a z1^2)^2 - 4 b (2 x1 + c z1) z1^3, p];
        f = Mod[4 z1 (x1^3 + c x1^2 z1 + a x1 z1^2 + b z1^3), p];
        Return[{e,f}]
];

ellodd[pt_, pu_, pv_] := Block[
                {x1 = pt[[1]], z1 = pt[[2]],
                 x2 = pu[[1]], z2 = pu[[2]],
                 xx = pv[[1]], zz = pv[[2]], i, j},
            i = Mod[zz ((x1 x2 - a z1 z2)^2 -
                4 b(x1 z2 + x2 z1 + c z1 z2) z1 z2), p];
            j = Mod[xx (x1 z2 - x2 z1)^2, p];
                 Return[{i,j}]
];

bitList[k_] := Block[{li = {}, j = k},
        While[j > 0,
          li = Append[li, Mod[j,2]];
          j = Floor[j/2];
        ];
        Return[Reverse[li]];
        ];

elliptic[pt_, k_] := Block[{porg, ps, pp, q},

        If[k ==1, Return[pt]];
        If[k ==2, Return[elleven[pt]]];
        porg = pt;
        ps = elleven[pt];
        pp = pt;
        bitlist = bitList[k];
```

```
        Do[
          If[bitlist[[q]] == 1,
                pp = ellodd[ps, pp, porg];
                ps = elleven[ps],
                  ps = ellodd[pp, ps, porg];
                  pp = elleven[pp]
          ],
          {q,2,Length[bitlist]}
      ];
      Return[Mod[pp,p]]
    ];
    ellinv[n_] := PowerMod[n,-1,p];
    ex[pt_] := Mod[pt[[1]] * ellinv[pt[[2]]], p];
    squareQ[x_] := PowerMod[x, (p-1)/2, p] != (p-1);
    pointQ[x_] := squareQ[x^3 + c x^2 + a x + b];


    (* Direct embedding algorithm (FEED). *)

    elladd[x1_, x2_, sgn_] := Block[{u2, v},
      If[x1 == x2, Return[
        Mod[(x1^2-a)^2 PowerMod[4(x1^3 + c x1^2 + a x1 + b), -1, p],p]
        ]];
      v = Mod[((x1 x2 - a)^2-4b(x1+x2+c)) ellinv[x1-x2]^2, p];
      u2 = Mod[ ((x1 x2 + a)(x1 + x2) + 2c x1 x2 + 2b) *
          ellinv[x1-x2]^2, p];
       Mod[u2 + sgn*
         PowerMod[u2^2 - v, (p+1)/4, p], p]
    ];

    q = 192; k = 1425;
    p = 2^q-k;
    c = 0; a = 0; b = -11;
    p1[1] = {841082007613983662909216085212018592355989658924032240952,1};
    p1[-1] = {303392091279366185250745192897508646125056720890157126474 4,1};
    aPri = 4434334;
    bPri = 418245599585;
    aPub[1] = elliptic[p1[1], aPri];
```

```
aPub[-1] = elliptic[p1[-1], aPri];
bPub[1] = elliptic[p1[1], bPri];
bPub[-1] = elliptic[p1[-1], bPri];

xp = 11111111333377; (* Plaintext. *)
curve = If[pointQ[xp], 1, -1];
Print[xp," ",curve];
(* next, test parcel with various random integers r. *)

Do[
        r = Random[Integer, 32767];
        xq = ex[elliptic[bPub[curve], r]];
        xm = elladd[xp, xq, +1];
        xc = ex[elliptic[p1[curve], r]];   (* Sender's clue. *)
        g = If[xp == elladd[xm, xq, +1], 1, -1];
        Print["Transmit: ",{xm, xc, g}];
        Print["Decrypt: ",
                elladd[xm, ex[elliptic[{xc,1}, bPri]], g]];
  ,{qq,1,9}
];
```

A function to compare signatures using the optimized scheme is as follows:

```
int
signature_compare(key p1, key p2, key p3);
/*  Returns non-zero if x(p1) cannot be the x-coordinate of the sum of
two points whose respective x-coordinates are x(p2),
x(p3). */
```

A function to calculate Q and compare it with (P + *M*(ciphertext, P)°ourPub) is as follows:

```
q = new_public_from_private (NULL, depth, seed);
```

```
elliptic_mul (q, u);  /* u is the random integer. */
elliptic_mul (our, m);  /* m = M(cipertext, P). */
/* Next, use the transmitted point p. */
if(signature_compare (p, our, q))
        fprintf(stderr,"Signature invalid.\n");
```

Encryption/Decryption

The encryption/decryption schemes of the present invention can be implemented in the programming language C. The following are examples of programmatic interfaces (.h files) and test programs (.c files) suitable for implementing the encryption/decryption of the present invention.

```
/* fee.h

   © 1991 NeXT Computer, Inc. All Rights Reserved.
 */

#import "giants.h"

#define DEFAULT_VERSION 1 #define DEFAULT_DEPTH 4 #define DEFAULT_SEED 0
#define MAX_DEPTH 22 #define FEE_TOKEN "scicompg" #define BUF_SIZE 8192
#define KEY_TOO_SHORT 1 #define ILLEGAL_CHARS_IN_KEY 2 #define BAD_TOKEN
3 #define VERSION_PARAM_MISMATCH 4 #define DEPTH_PARAM_MISMATCH 5
#define SEED_PARAM_MISMATCH 6 #define EXP_PARAM_MISMATCH 7 #define
A_PARAM_MISMATCH 8 #define X1_PARAM_MISMATCH 9

typedef giant padkey;

typedef struct {
          int version; int depth; int seed; int exp; int a; int x1;
          padkey x;
} keystruct; typedef keystruct *key;

int hexstr_illegal(char *pub_hex); /* Returns non-zero iff pub_hex is
not a valid hex string. */
```

```
void hexstr_to_key(char *str, key public); /* Jams public (assumed pre-
malloced) with hex str contents. */

char * new_hexstr_from_key(key public); /* Mallocs and returns a hex
string representing public. */

key new_public_from_private(char *private, int depth, int seed); /*
Mallocs and returns a new public key. If private==NULL, depth and seed
are ignored, and the returned key is simply malloc'ed but without
meaningful parameters. If private is a valid string, depth and seed are
used to establish correct elliptic parameters. depth is 0 to MAx_DEPTH
inclusive, while seed = DEFAULT_SEED usually, but may be chosen to be
any integer in order to change the encryption parameters for the given
depth. The depth alone determines the time to generate one-time pads.
 */

char * new_hexstr_from_pad(); /* Malloc's and returns a hex string,
null-terminated, representing the one-time pad. This function is usually
called after a make_one_time_pad() call.
 */

void generate_byte_pad(char *byte_pad, int len); /* Jams byte_pad with
len bytes of the one-time pad. There is no null termination; just len
bytes are modified.
 */

int make_one_time_pad(char *private, key public); /* Calculate the
internal one-time pad. */

void free_key(key pub); /* De-allocate an allocated key. */

void NXWritePublic(NXStream *out, key my_pub); /* Write a key to out
stream. */

void NXReadPublic(NXStream *in, key pub); /* Read a key from in stream.
*/

int keys_inconsistent(key pub1, key pub2); /* Return non-zero if pub1,
pub2 have inconsistent parameters.
 */

int encrypt_stream(NXStream *in, NXStream *out, key their_pub, key
my_pub, char *my_pri); /* Encrypt in to out. If my_pub!=NULL, a
consistency check for equivalent parameters with their_pub is performed,
```

```
with possible non-zero error returned (and encryption aborted).
Otherwise, when my_pub==NULL, an internal key is temporarily created for
insertion into the out stream.
 */

int decrypt_stream(NXStream *in, NXStream *out, char *my_pri); /*
Decrypt in to out. Non-zero error value is returned if an internal token
(that should have been present in the in stream) is not properly
decrypted.
 */

void set_crypt_params(int *depth, int *exp, int *a, int *x1, int *seed);

void str_to_giant(char *str, giant g);

int ishex(char *s);

void byte_to_hex(int b, char *s);

void hex_to_byte(char *s, int *b);

int hexstr_to_int(char **s);

int int_to_hexstr(int n, char *str);

int giant_to_hexstr(giant g, char *str);

void make_base(int exp);

void init_elliptic();

padkey get_pad();

void ell_even(giant x1, giant z1, giant x2, giant z2, int a, int q);

void ell_odd(giant x1, giant z1, giant x2, giant z2, giant xor, giant
zor, int q);

int scompg(int n, giant g);

void elliptic(giant xx, giant zz, giant k, int a, int q);

unsigned char byt(padkey x, int k);

int version_param(key pub);
```

```
int depth_param(key pub);
int seed_param(key pub);
int exp_param(key pub);
int a_param(key pub);
int x1_param(key pub);
```

```
/* keytest.c
   Test program for public key exchange, Usage: > keytest depth
   MyPrivate TheirPrivate

   © 1991 NeXT Computer, Inc. All Rights Reserved
 */

#import <stdio.h> #import <streams/streams.h> #import "fee.h"

main(int argc, char **argv) {
        key my_pub, their_pub; char *my_pub_str, *their_pub_str; char
        *padstr; int depth;

        if(argc<4) {
                fprintf(stderr, "Usage: keytest depth MyPrivate
                TheirPrivate\n"); exit(0);
        }

        depth = atoi(argv[1]); my_pub =
        new_public_from_private(argv[2], depth, DEFAULT_SEED);
        their_pub = new_public_from_private(argv[3], depth,
        DEFAULT_SEED);

        my_pub_str = new_hexstr_from_key(my_pub); their_pub_str =
        new_hexstr_from_key(their_pub);

        printf("My Public Key:\n%s\n",my_pub_str); printf("Their
        Public Key:\n%s\n",their_pub_str);

        free(my_pub_str); free(their_pub_str);

        make_one_time_pad(argv[2], their_pub); padstr =
        new_hexstr_from_pad(); printf("One-time pad, using My Private
        and Their Public:\n%s\n",padstr); free(padstr);

        make_one_time_pad(argv[3], my_pub); padstr =
        new_hexstr_from_pad(); printf("One-time pad, using Their
        Private and My Public:\n%s\n",padstr); free(padstr);

        free_key(my_pub); free_key(their_pub);

        printf("The two one-time pads should be equivalent.\n");
```

```
/* solencrypt.c
   Solitaire encryption for personal files, Usage: > solencrypt <depth>
   file file.ell Private Key:

   © 1991 NeXT Computer, Inc. All Rights Reserved
 */

#import <stdio.h> #import <streams/streams.h> #import "fee.h"

main(int argc, char **argv) {
        key my_pub; int depth; char *my_pri; NXStream *inStream,
        *outStream;

        if(argc<3) {
fprintf(stderr, "Usage: solencrypt <depth> file file.ell\nPrivate Key:
\nwhere depth is an integer 0 through 22, def ault = 4.\n");
        exit(0); } if(argc==4) depth = atoi(argv[1]); else depth =
        DEFAULT_DEPTH;

/* Next, open the streams. */

        inStream = NXMapFile(argv[argc-2],NX_READONLY); outStream =
        NXOpenMemory(NULL,0,NX_WRITEONLY);

/* Next, get private key, make public key, encrypt stream, blank the
private key in memory. */

        my_pri = (char *) getpass("Private Key: "); my_pub =
        new_public_from_private(my_pri, depth, DEFAULT_SEED);
        encrypt_stream(inStream, outStream, my_pub, my_pub, my_pri);
        bzero(my_pri, strlen(my_pri)); free_key(my_pub);

/* Next, flush and write. */

        NXFlush(inStream); NXFlush(outStream); NXSaveToFile(outStream,
        argv[argc-1]); NXClose(inStream); NXCloseMemory(outStream,
        NX_FREEBUFFER);
```

```
/* soldecrypt.c
   Solitaire encryption for personal files, Usage: > soldecrypt file.ell
   file Private Key:

   © 1991 NeXT Computer, Inc. All Rights Reserved
 */

#import <stdio.h> #import <streams/streams.h> #import "fee.h"

main(int argc, char **argv) {
        char *my_pri; NXStream *inStream, *outStream; int err;

        if(argc<3) {
                fprintf(stderr, "Usage: soldecrypt file.ell
                file\nPrivate Key: \n"); exit(0);
        }

/* Next, open the streams. */

        inStream = NXMapFile(argv[1],NX_READONLY); outStream =
        NXOpenMemory(NULL,0,NX_WRITEONLY);

/* Next, decrypt the stream and blank the private key in memory. */

        my_pri = (char *) getpass("Private Key: "); err =
        decrypt_stream(inStream, outStream, my_pri); bzero(my_pri,
        strlen(my_pri)); if(err) {
                fprintf(stderr,"Error %d: bad private key.\n", err);
                exit(0);
        }

/* Next, write and close. */

        NXSaveToFile(outStream, argv[2]); NXClose(inStream);
        NXCloseMemory(outStream, NX_FREEBUFFER);
```

CLAIMS OF THE INVENTION

1. A method for encrypting a plaintext message in a sender computer system comprising the steps of:

selecting a parcel of plaintext x_{text};

determining for which of two elliptic curves E^+ and E^- x_{text} is a valid coordinate;

generating a message coordinate x_m using a random value r, a public key from a public key/private key pair, and x_{text};

generating a clue value x_c;

generating a sign value g;

representing said encrypted message by said message coordinate, said clue, and said sign.

2. The method of claim 1 wherein said message coordinate, said clue, and said sign are transmitted to a receiver.

3. The method of claim 2 wherein said public key is a public key of said receiver.

4. The method of claim 1 wherein said step of determining for which of two curves x_{text} is a valid coordinate is accomplished by the following steps:

assuming q, a=0, b=±1, c≠2, giving rise to two possible elliptic curves $E^{\pm}$;

assuming two public coordinates $x_1^{\pm}$ such that two points P_1^+ and P_1^- lie respectively on curves E^+ and E^-;

determining the sign of the curve for which x_{text} is a valid coordinate by

$(x_{text}Q(x_{text}))^{(p+1)/2} = \pm x_{text}Q(x_{text}) \pmod p$

$p = 3 \pmod 4$.

5. The method of claim 4 wherein the step of generating a message coordinate is accomplished by the following steps:

choosing random r, and calculating $(x_q, ?) := r * (\text{theirPub}^{\pm}, ?)$;

calculating a message coordinate $x_m := \text{elliptic\_add}(x_{text}, x_q, +1)$

6. The method of claim 5 wherein said step of generating a clue x_c is accomplished by $(x_c, ?) := r \cdot P_1^{\pm}$.

7. The method of claim 6 wherein said step of determining said sign g is accomplished by which sign holds in elliptic_add $(x_m, x_q, +1) = x_{text}$.

8. The method of claim 7 further including the steps of:

at said receiver, generating x_q from said clue x_c;

recovering the plaintext by x_{text} := elliptic_add (x_m, x_c, g).

9. An article of manufacture comprising:

a computer usable medium having computer readable program code embodied therein for encrypting a plaintext message using elliptic curve algebra, the computer readable program code in said article of manufacture comprising;

computer readable program code configured to cause a computer to select a parcel of plaintext x_{text};

computer readable program code configured to cause a computer to determine for which of two elliptic curves E^+ and E^- x_{text} is a valid coordinate;

computer readable program code configured to cause a computer to generate a message coordinate x_m using a random value r, a public key from a public key/private key pair, and x_{text};

computer readable program code configured to cause a computer to generate a clue value x_c;

computer readable program code configured to cause a computer to generate a sign value g;

computer readable program code configured to cause a computer to represent said encrypted message by said message coordinate, said clue, and said sign.

10. The article of manufacture of claim 9 wherein said message coordinate, said clue, and said sign are transmitted to a receiver.

11. The article of manufacture of claim 10 wherein said public key is a public key of said receiver.

12. The article of manufacture of claim 9 wherein said computer readable program code configured to cause a computer to determine for which of two curves x_{text} is a valid coordinate comprises computer readable program code configured to cause a computer to perform the following steps of:

assuming q, $a=0$, $b=\pm 1$, $c\neq 2$, giving rise to two possible elliptic curves $E^{\pm}$;

assuming two public coordinates $x_1^{\pm}$ such that two points P_1^+ and P_1^- lie respectively on curves E^+ and E^-;

determining the sign of the curve for which x_{text} is a valid coordinate by

$$(x_{text}Q(x_{text}))^{(p+1)/2} = \pm x_{text}Q(x_{text}) \pmod{p}$$

$$p = 3 \pmod 4.$$

13. The article of manufacture of claim 12 wherein the computer readable program code configured to cause a computer to generate a message coordinate comprises:

computer readable program code configured to cause a computer to choose random r, and calculate $(x_q, ?) := r * (\mathbf{theirPub}^{\pm}, ?)$;

computer readable program code configured to cause a computer to calculate a message coordinate x_m := elliptic_add(x_{text}, x_q, +1)

14. The article of manufacture of claim 13 wherein said computer readable program code configured to cause a computer to generate a clue x_c comprises computer readable program code configured to cause a computer to compute $(x_c, ?) := r \cdot P_1^{\pm}$.

15. The article of manufacture of claim 14 wherein said computer readable program code configured to cause a computer to determine said sign g comprises computer readable program code configured to cause a computer to determine which sign holds in elliptic_add $(x_m, x_q, +1) = x_{text}$.

16. The article of manufacture of claim 15 further including:

computer readable program code configured to cause a computer to, at said receiver, generate x_q from said clue x_c;

computer readable program code configured to cause a computer at said receiver to recover the plaintext by x_{text} := elliptic_add (x_m, x_c, g).

17. A method for encrypting a plaintext message comprising the steps of:

selecting two random numbers r and s;

generating an initial clue $clue_0$ using said random number r, a public key from a first public key/private key pair, and a private key from a second public key/private key pair;

selecting a parcel of plaintext x_{text_i};

determining for which of two elliptic curves E^+ and E^- x_{text_i} is a valid coordinate;

generating a message coordinate m_i using x_{text_i} and a current clue $clue_i$;

generating a sign value g;

representing said encrypted message by the pair (m_i, g); and

for a subsequent parcel, generating a subsequent clue $clue_{i+1}$ using said current clue $clue_i$ and said random numbers r and s.

18. The method of claim 17 further comprising the steps of:

transmitting said random numbers r and s from a sender to a receiver; and

transmitting said encrypted message from said sender to said receiver.

19. The method of claim 18 wherein said public key is a public key of said receiver and said private key is a private key of said sender.

20. The method of claim 17 wherein said step of generating said initial clue $clue_0$ comprises the step of computing:

$clue_0 = r \circ ourPri \circ theirPub^{\pm}$

where *ourPri* comprises said private key and $theirPub^{\pm}$ comprises said public key.

21. The method of claim 17 wherein said step of determining for which of two elliptic curves E^+ and E^- x_{text_i} is a valid coordinate comprises the steps of:

assuming q, $a=0$, $b=\pm 1$, $c \neq 2$, giving rise to two possible elliptic curves $E\pm$;

assuming two public coordinates $x_1^{\pm}$ such that two points P_1^+ and P_1^- lie respectively on curves E^+ and E^-; and

determining the sign of the curve for which x_{text_i} is a valid point by

$$(x_{text_i}Q(x_{text_i}))^{(p+1)/2} = \pm x_{text_i}Q(x_{text_i}) \pmod{p}$$

$$p = 3 \pmod 4.$$

22. The method of claim 17 wherein the step of generating said message coordinate comprises the step of computing

m_i := elliptic_add(x_{text_i}, $clue_i$, +1).

23. The method of claim 17 wherein said step of generating said sign g comprises the step of determining which sign recovers x_{text_i} when elliptic_add(m_i, $clue_i$, ± 1) is computed.

24. The method of claim 17 wherein said step of generating said subsequent clue $clue_{i+1}$ comprises the steps of:

computing $s = s \circ P^{\pm}$; and

computing $clue_{i+1}$ = elliptic_add($r \circ clue_i$, s, +1).

25. The method of claim 18 further comprising the following steps performed at said receiver:

determining said initial clue $clue_0$ from said random number r, a private key of said first public key/private key pair, and a public key of said second public key/private key pair;

determining which elliptic curve holds the point m_i;

computing elliptic_add($clue_i$, m_i, g) to determine x_{text_i}; and

computing subsequent clue $clue_{i+1}$ using current clue $clue_i$ and said random numbers r and s.

26. An article of manufacture comprising:

a computer usable medium having computer readable program code embodied therein for causing a computer to encrypt a plaintext message using elliptic curve algebra, said computer readable program code comprising:

computer readable program code configured to cause a computer to select two random numbers r and s;

computer readable program code configured to cause a computer to generate an initial clue $\mathbf{clue}_0$ using said random number r, a public key from a first public key/private key pair, and a private key from a second public key/private key pair;

computer readable program code configured to cause a computer to select a parcel of plaintext x_{text_i};

computer readable program code configured to cause a computer to determine for which of two elliptic curves E^+ and E^- x_{text_i} is a valid coordinate;

computer readable program code configured to cause a computer to generate a message coordinate m_i using x_{text_i} and a current clue $\mathbf{clue}_i$;

computer readable program code configured to cause a computer to generate a sign value g;

computer readable program code configured to cause a computer to represent said encrypted message by the pair (m_i, g); and

computer readable program code configured to cause a computer to generate, for a subsequent parcel, a subsequent clue $clue_{i+1}$ using said current clue $clue_i$ and said random numbers r and s.

27. The article of manufacture of claim 26 further comprising:

computer readable program code configured to cause a computer to transmit said random numbers r and s from a sender to a receiver; and

computer readable program code configured to cause a computer to transmit said encrypted message from said sender to said receiver.

28. The article of manufacture of claim 27 wherein said public key is a public key of said receiver and said private key is a private key of said sender.

29. The article of manufacture of claim 26 wherein said computer readable program code configured to cause a computer to generate said initial clue $clue_0$ comprises computer readable program code configured to cause a computer to compute:

$$clue_0 = r \circ ourPri \circ theirPub^{\pm}$$

where *ourPri* comprises said private key and $theirPub^{\pm}$ comprises said public key.

30. The article of manufacture of claim 26 wherein said computer readable program code configured to cause a computer to determine for which of two elliptic curves E^+ and E^- x_{text_i} is a valid coordinate comprises computer readable program code configured to cause a computer to:

assume q, a=0, b=±1, $c \neq 2$, giving rise to two possible elliptic curves $E\pm$;

assume two public coordinates $x_1^{\pm}$ such that two points P_1^+ and P_1^- lie respectively on curves E^+ and E^-; and

determine the sign of the curve for which x_{text_i} is a valid coordinate by

$$(x_{text_i}Q(x_{text_i}))^{(p+1)/2} = \pm x_{text_i}Q(x_{text_i}) \pmod{p}$$

$p = 3 \pmod 4$.

31. The article of manufacture of claim 26 wherein said computer readable program code configured to cause a computer to generate said message coordinate comprises computer readable program code configured to cause a computer to compute m_i := elliptic_add(x_{text_i}, $clue_i$, +1).

32. The article of manufacture of claim 26 wherein said computer readable program code configured to cause a computer to generate said sign g comprises computer readable program code configured to cause a computer to determine which sign recovers x_{text_i} when elliptic_add(m_i, $clue_i$, ±1) is computed.

33. The article of manufacture of claim 26 wherein said computer readable program code configured to cause a computer to generate said subsequent clue $clue_{i+1}$ comprises:

computer readable program code configured to cause a computer to compute $s = s \circ P^{\pm}$; and

computer readable program code configured to cause a computer to compute $clue_{i+1}$ = elliptic_add($r \circ clue_i$, s, +1).

34. The article of manufacture of claim 27 further comprising computer readable program code configured to cause a computer at said receiver to perform the following steps of:

determining said initial clue $clue_0$ from said random number r, a private key of said first public key/private key pair, and a public key of said second public key/private key pair;

determining which elliptic curve holds the point m_i;

computing elliptic_add($clue_i$, m_i, g) to determine x_{text_i}; and

computing subsequent clue $clue_{i+1}$ using current clue $clue_i$ and said random numbers r and s.

FIG. 1
PRIOR ART

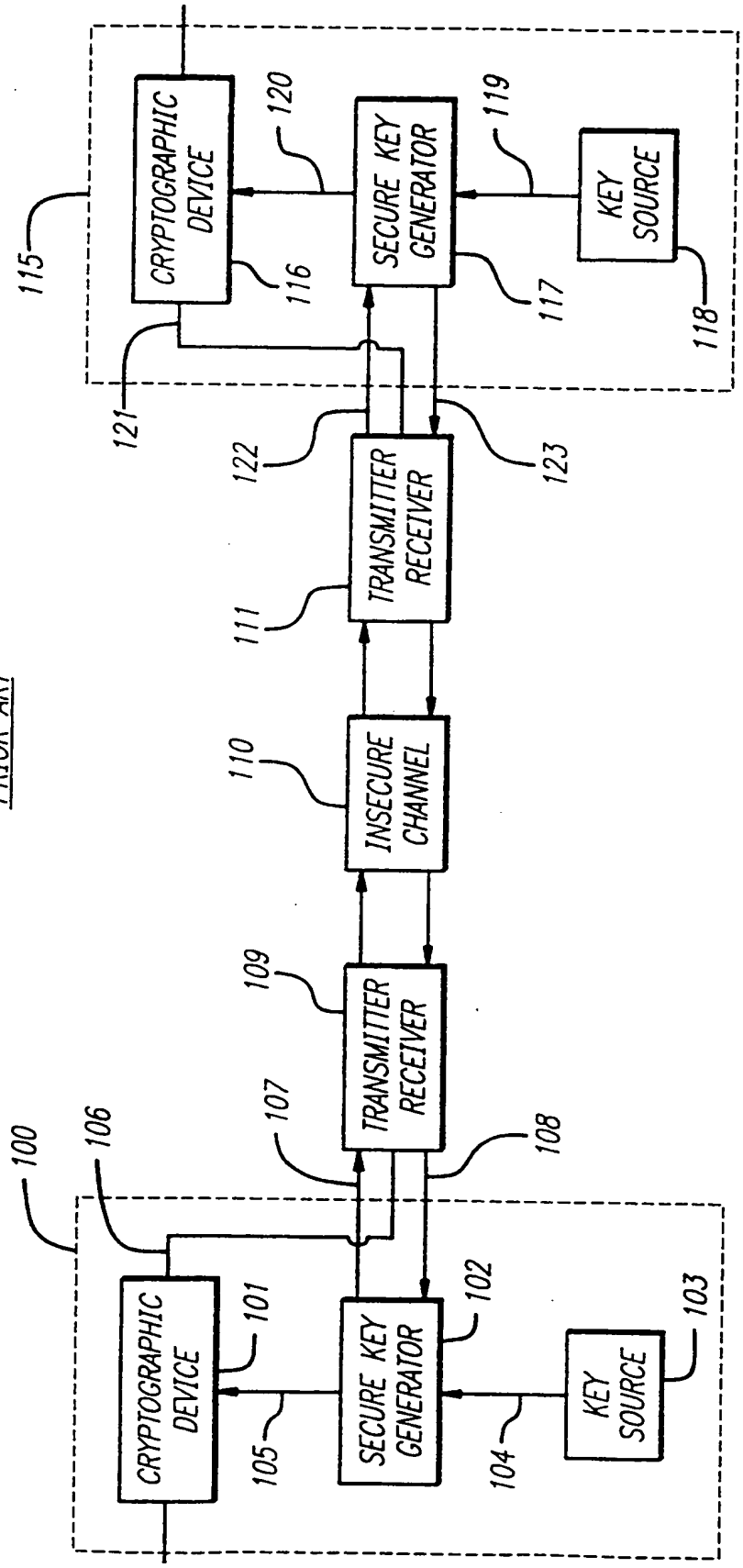


FIG. 2

PRIOR ART

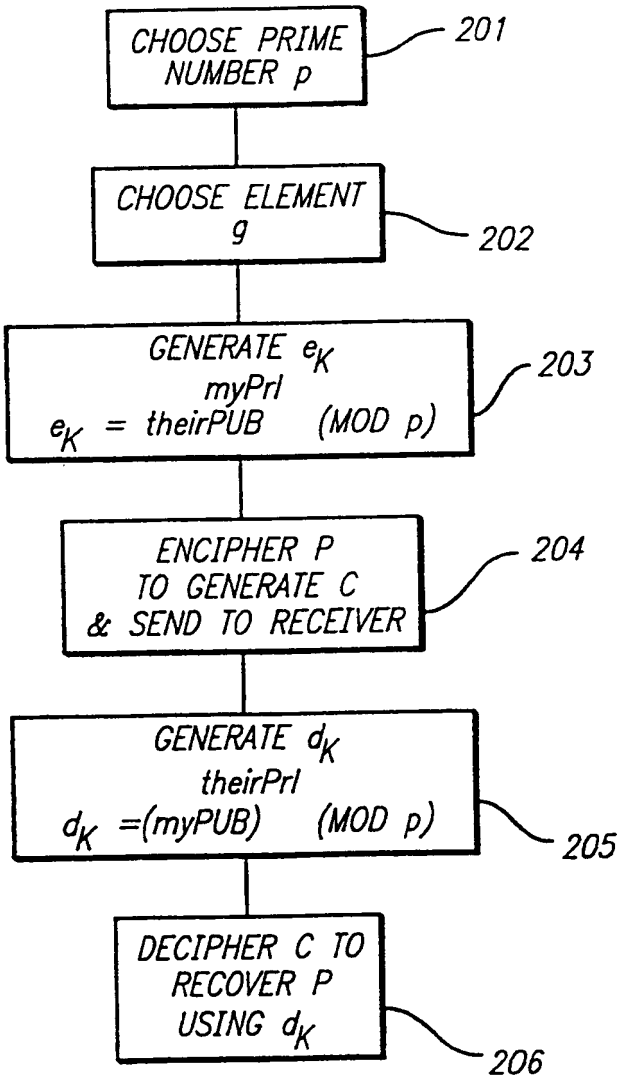


FIG. 3

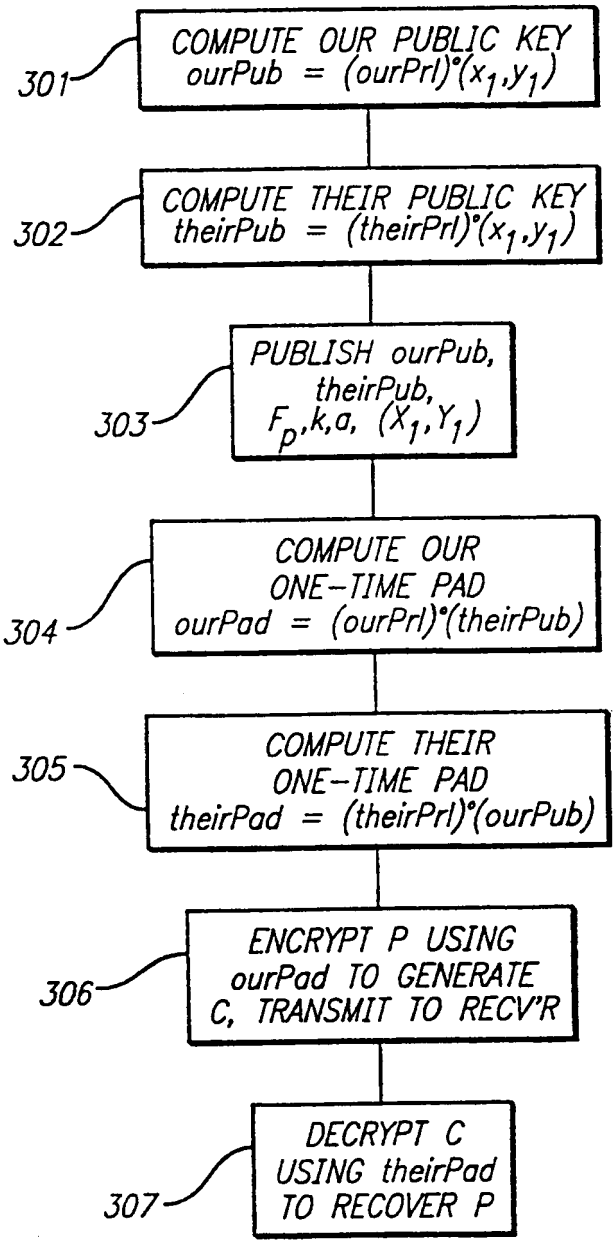


FIG. 4

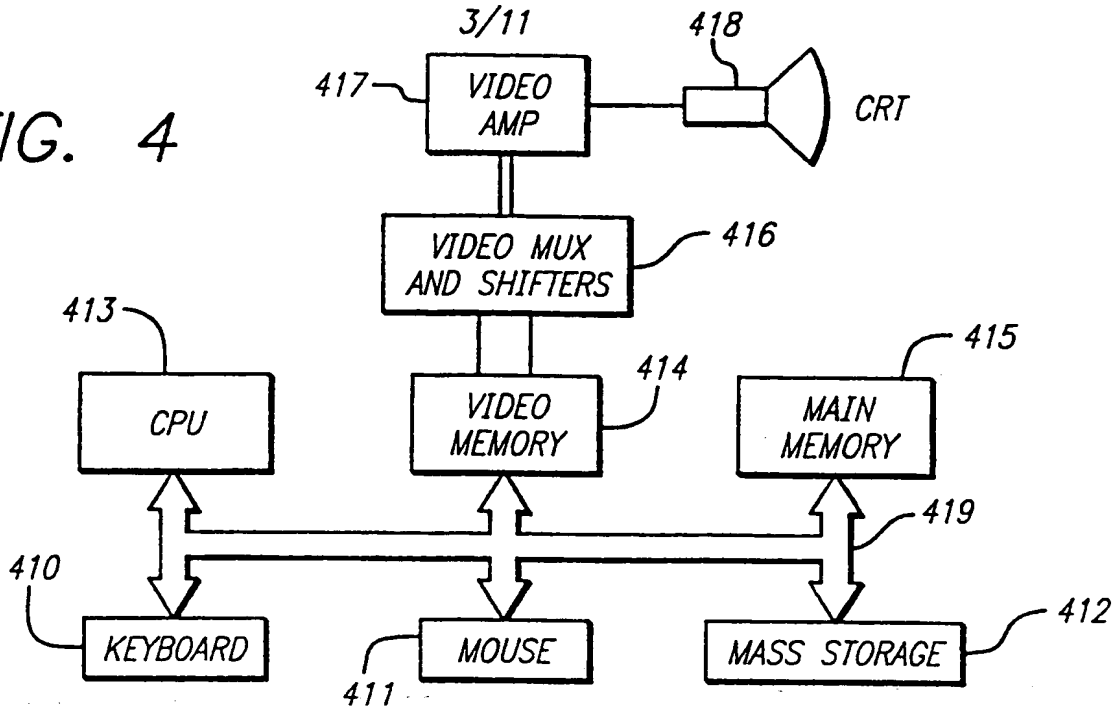
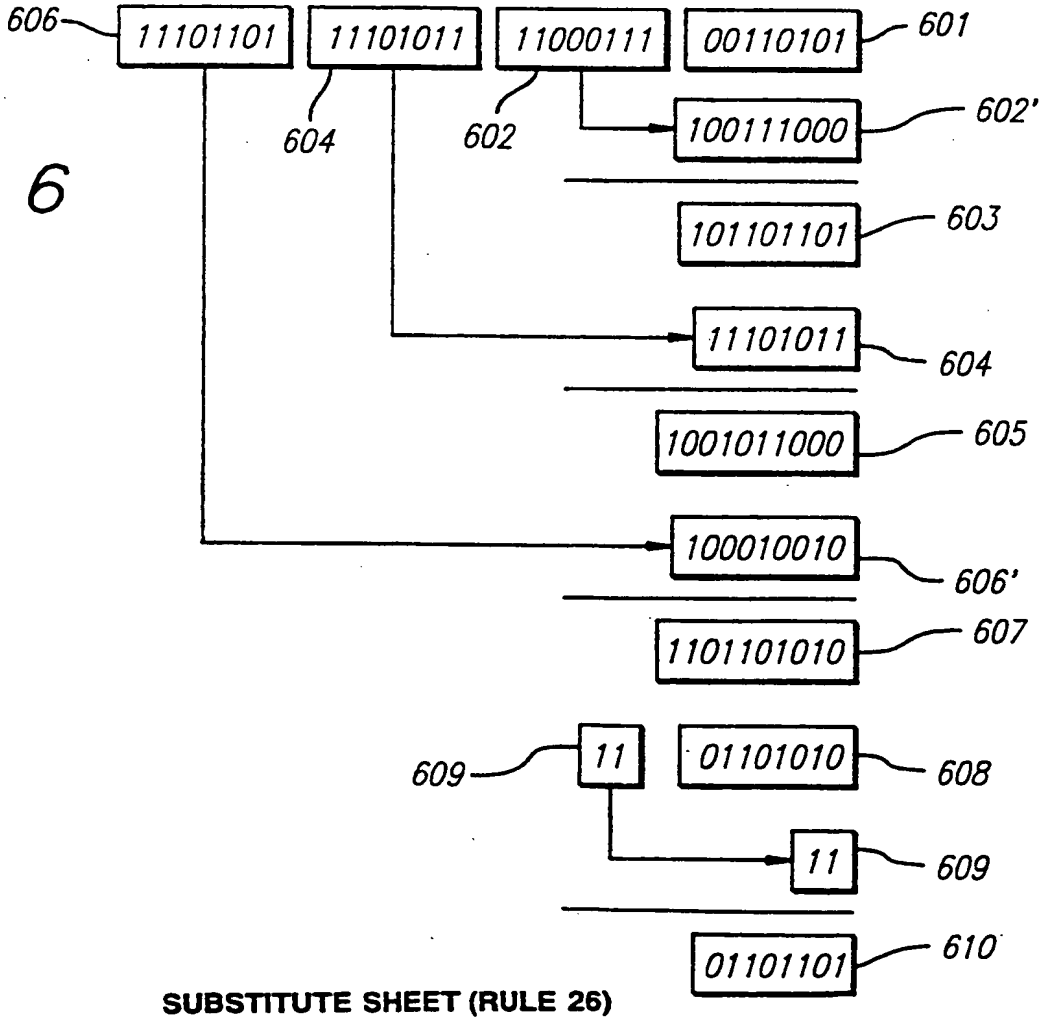


FIG. 6



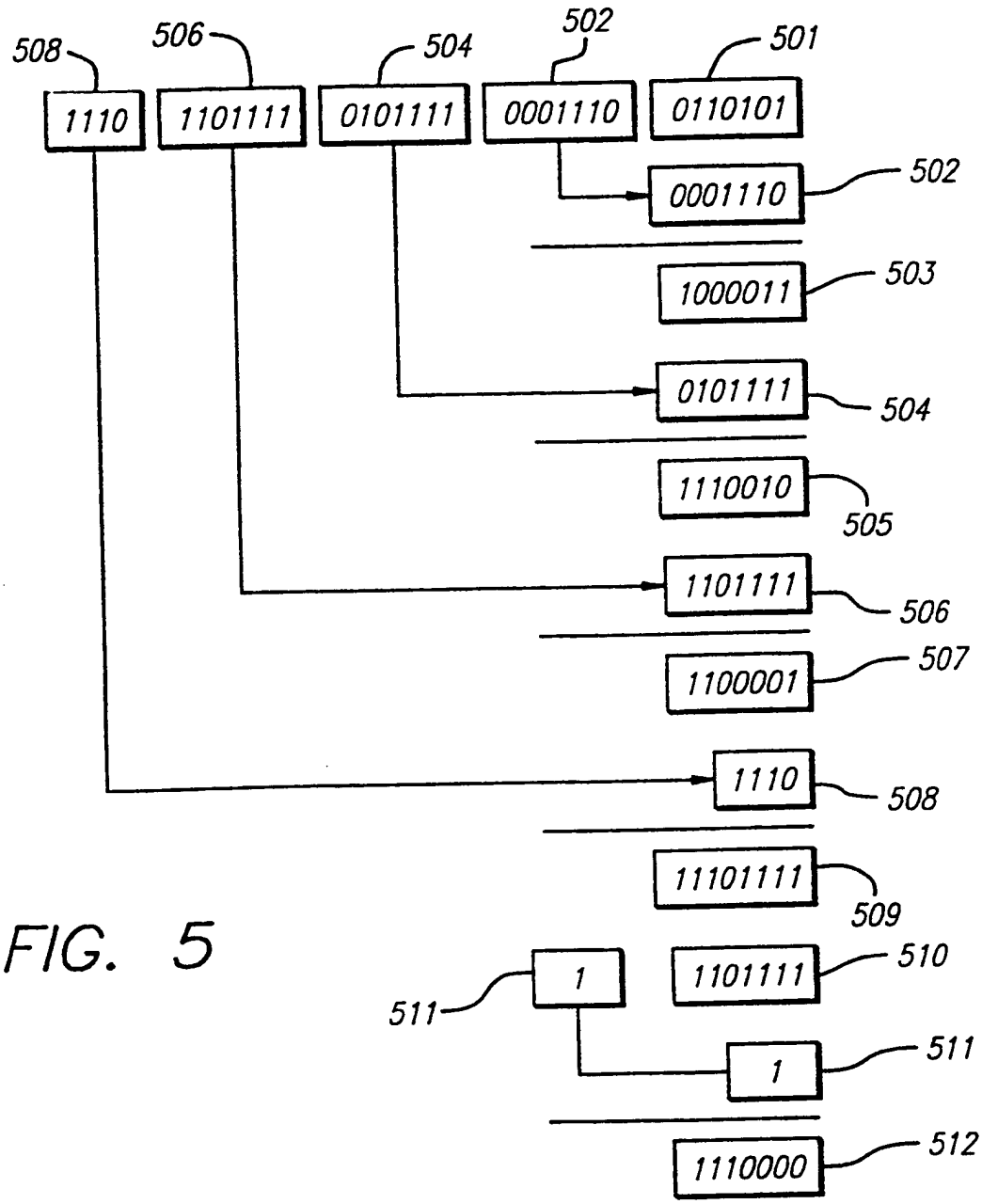

508
506
504
502
501
1110
1101111
0101111
0001110
0110101
0001110
502
1000011
503
0101111
504
1110010
505
1101111
506
1100001
507
1110
508
11101111
509
1
1101111
510
511
1
511
1110000
512

FIG. 5

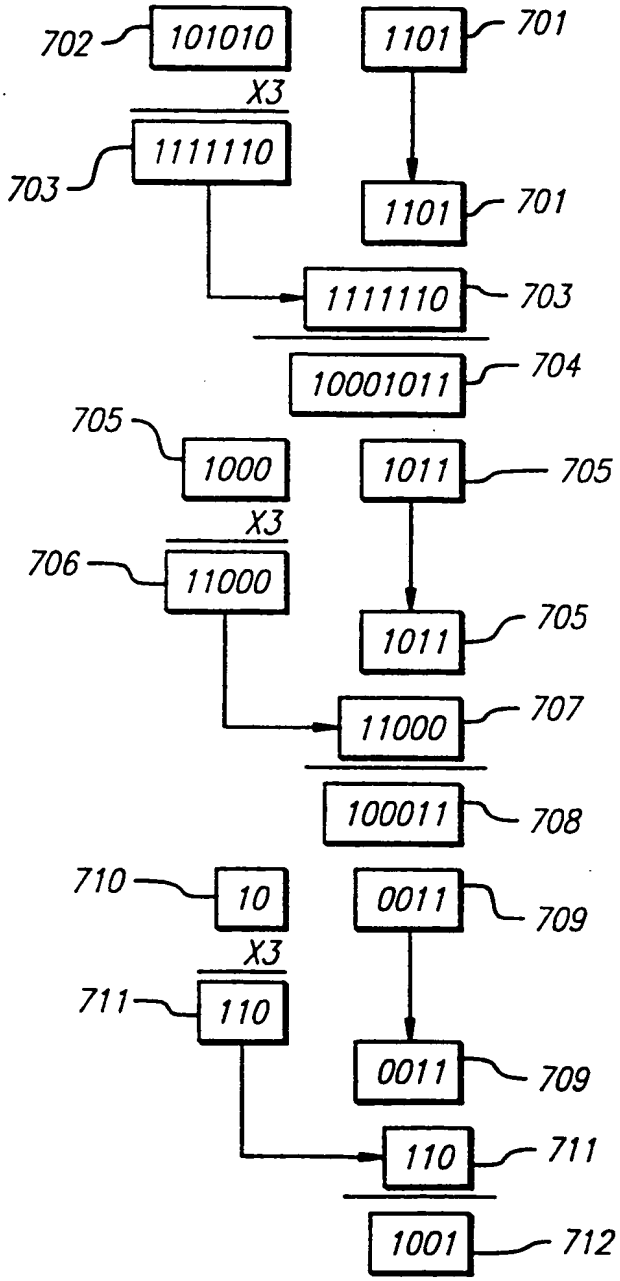


FIG. 7

FIG. 8

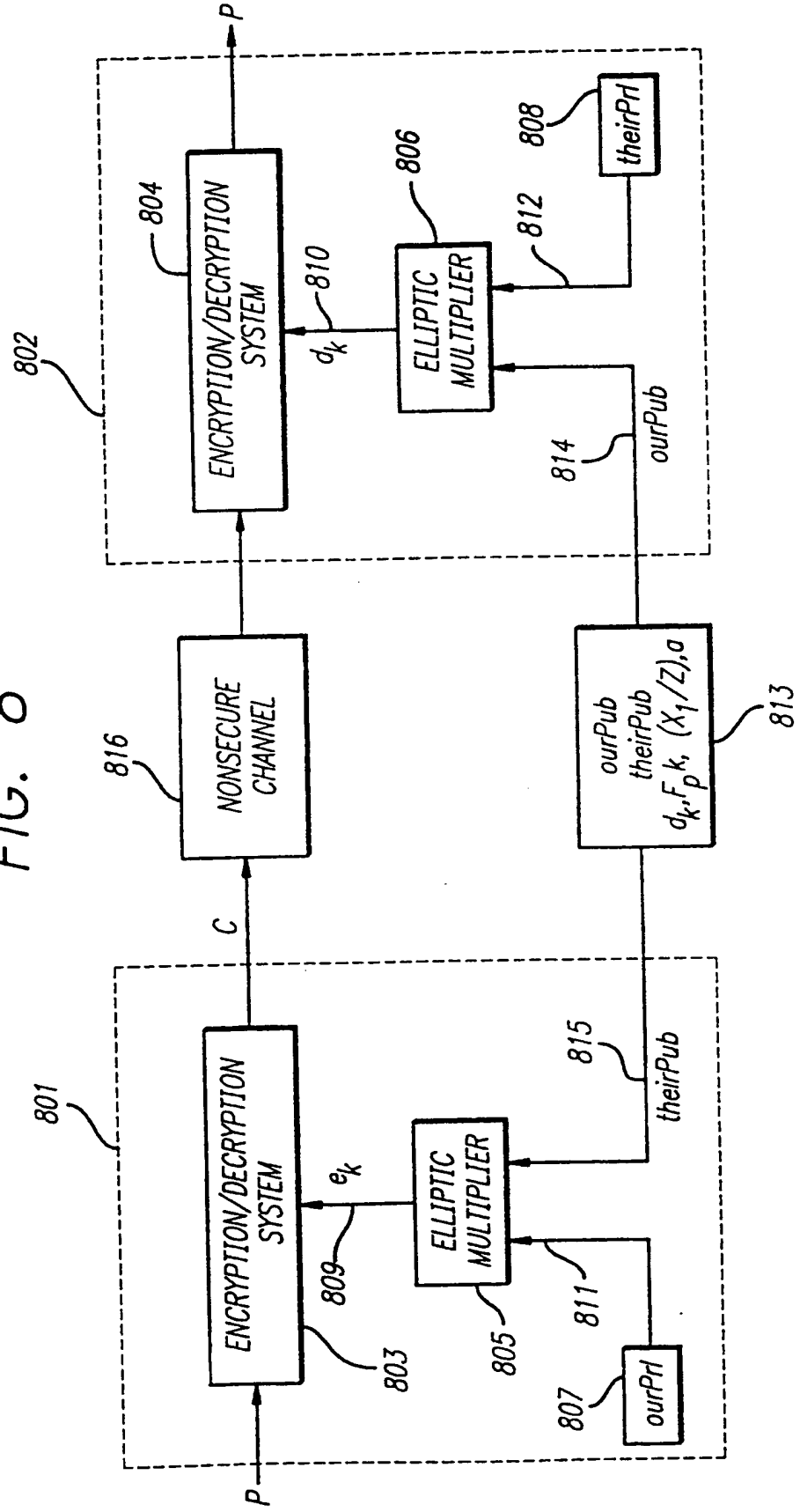


FIG. 9

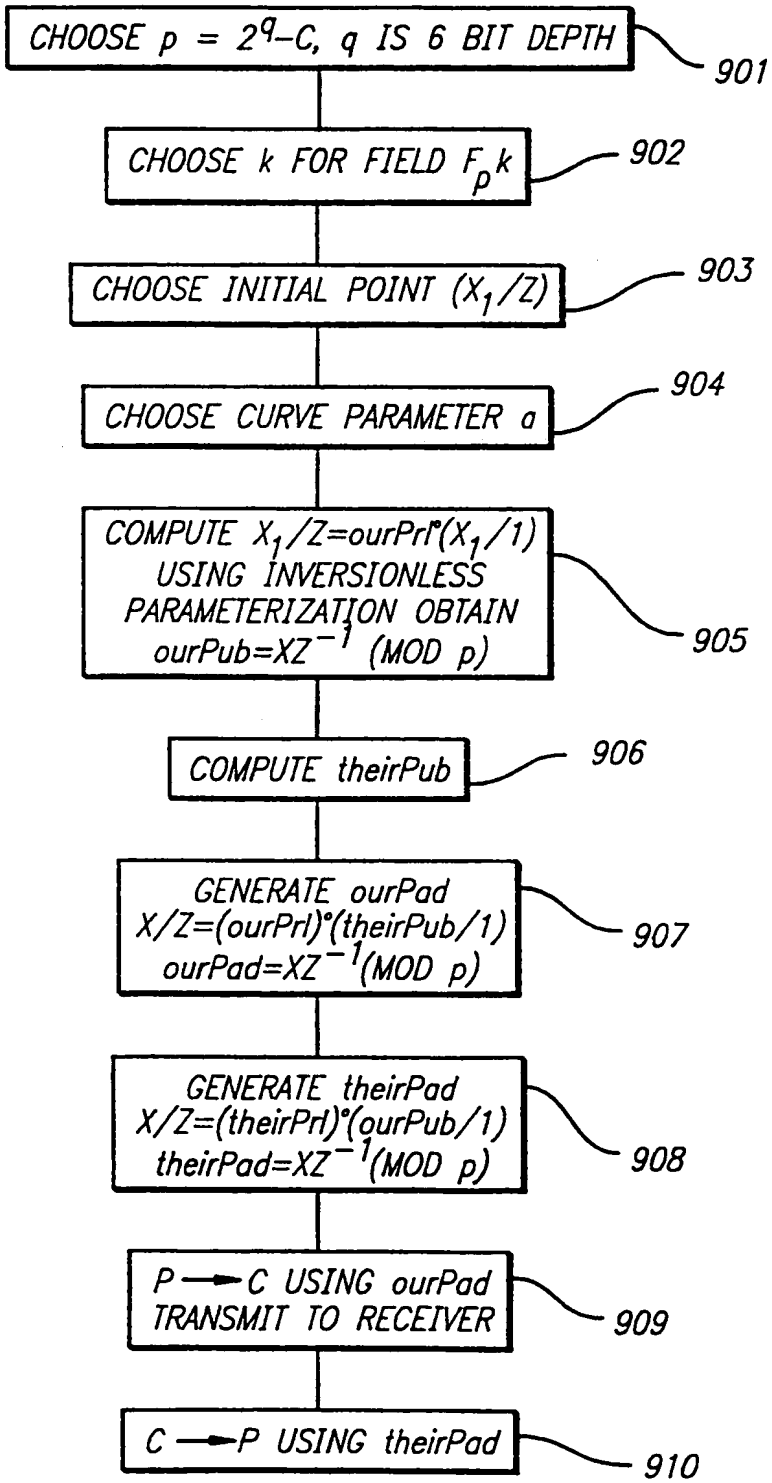


FIG. 10

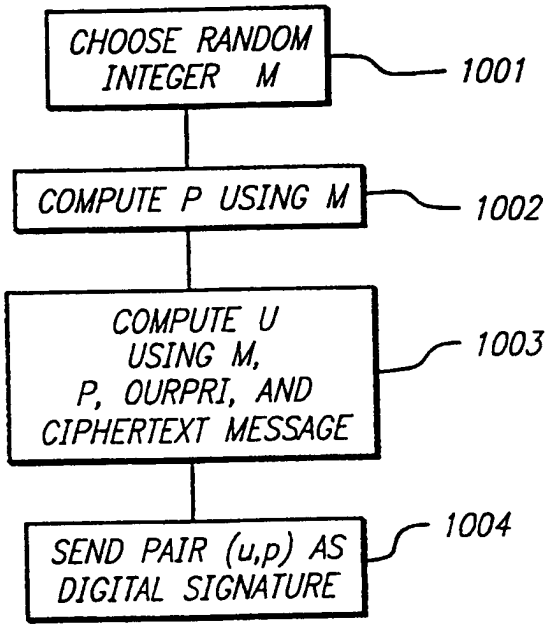
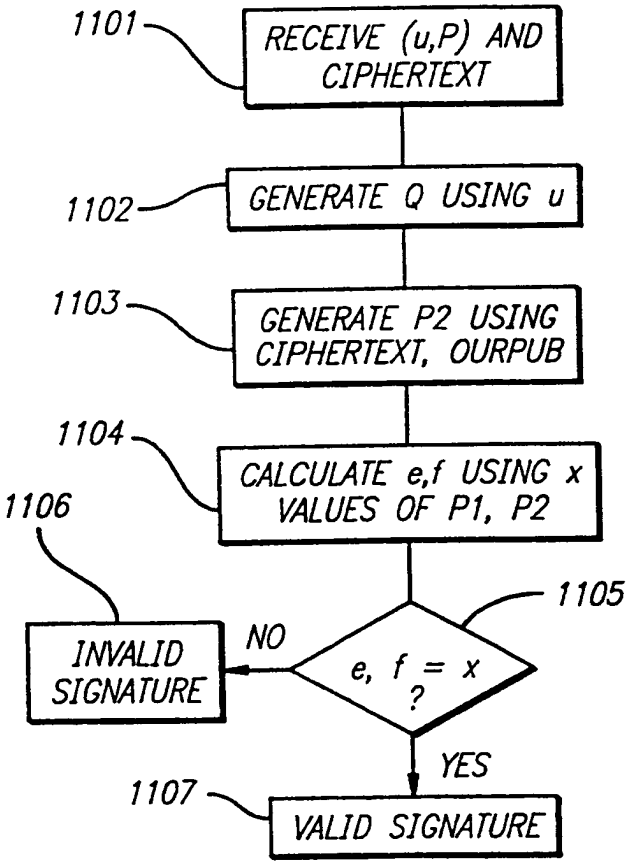


FIG. 11



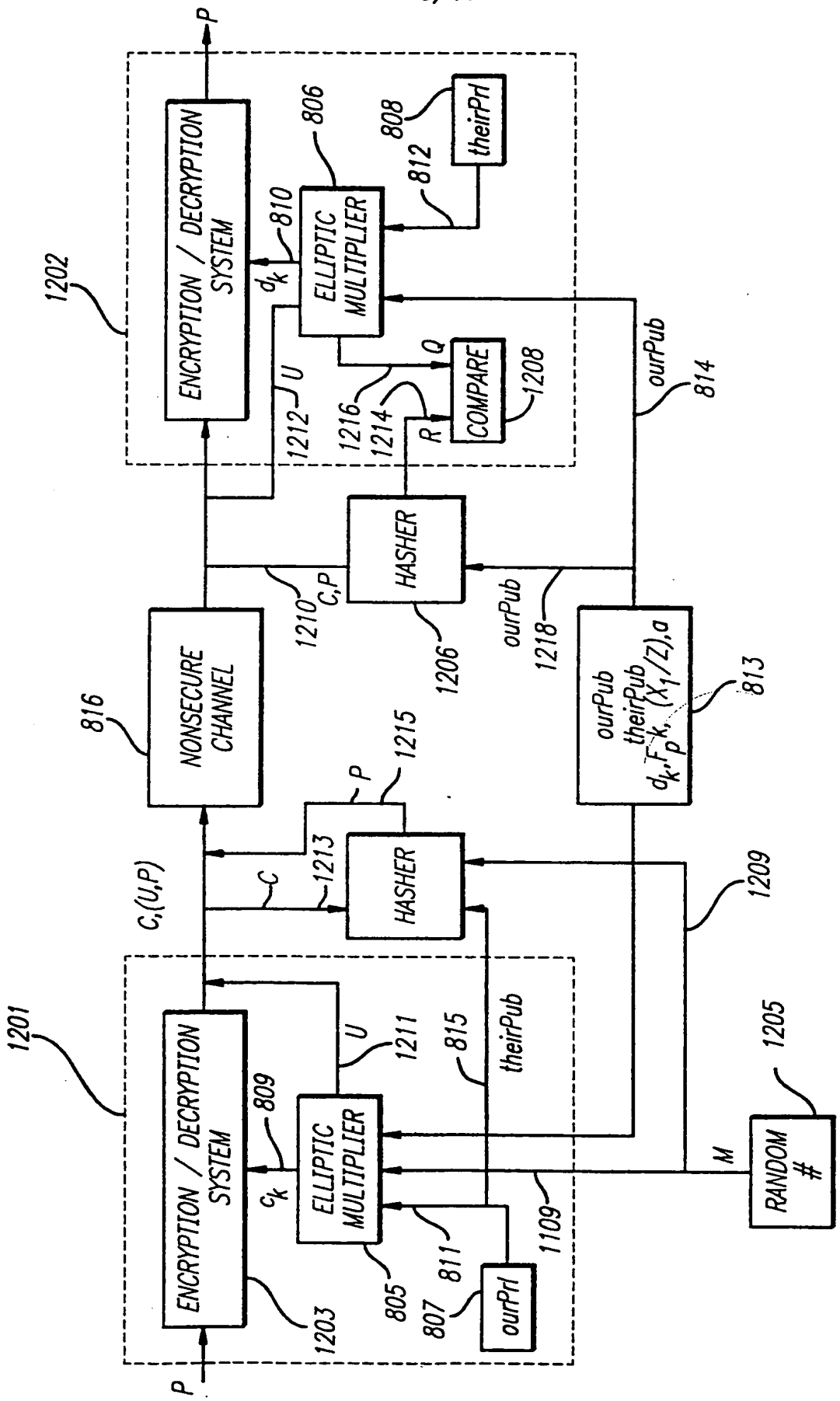


FIG. 12

FIG. 13

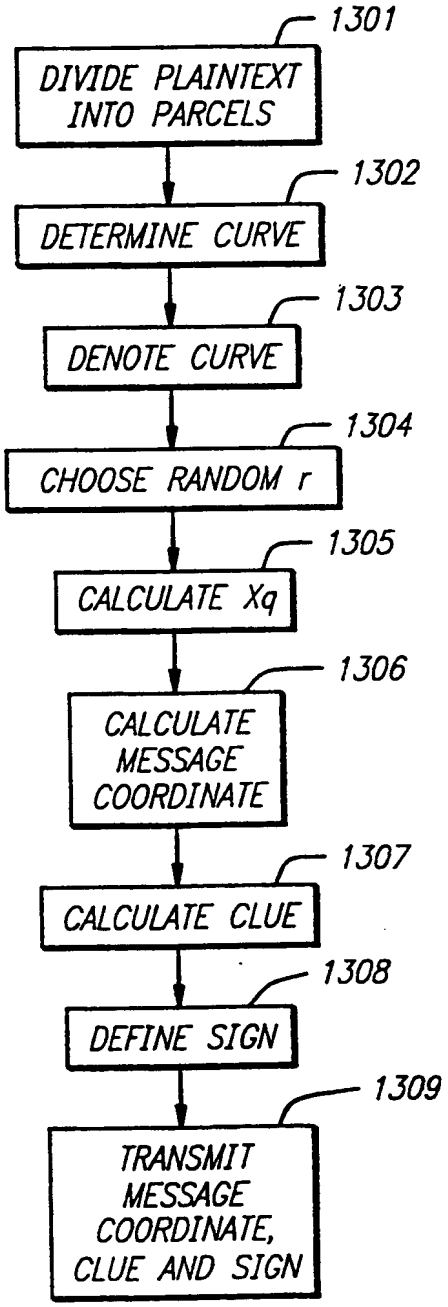

1301
DIVIDE PLAINTEXT INTO PARCELS
1302
DETERMINE CURVE
1303
DENOTE CURVE
1304
CHOOSE RANDOM r
1305
CALCULATE Xq
1306
CALCULATE MESSAGE COORDINATE
1307
CALCULATE CLUE
1308
DEFINE SIGN
1309
TRANSMIT MESSAGE COORDINATE, CLUE AND SIGN

FIG. 14

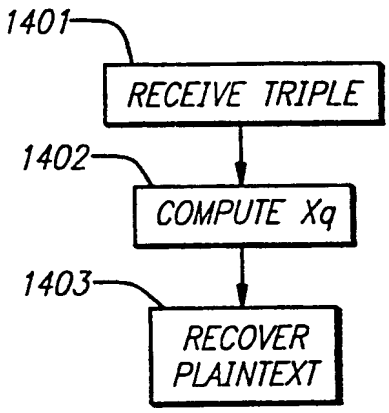

1401
RECEIVE TRIPLE
1402
COMPUTE Xq
1403
RECOVER PLAINTEXT

FIG. 15

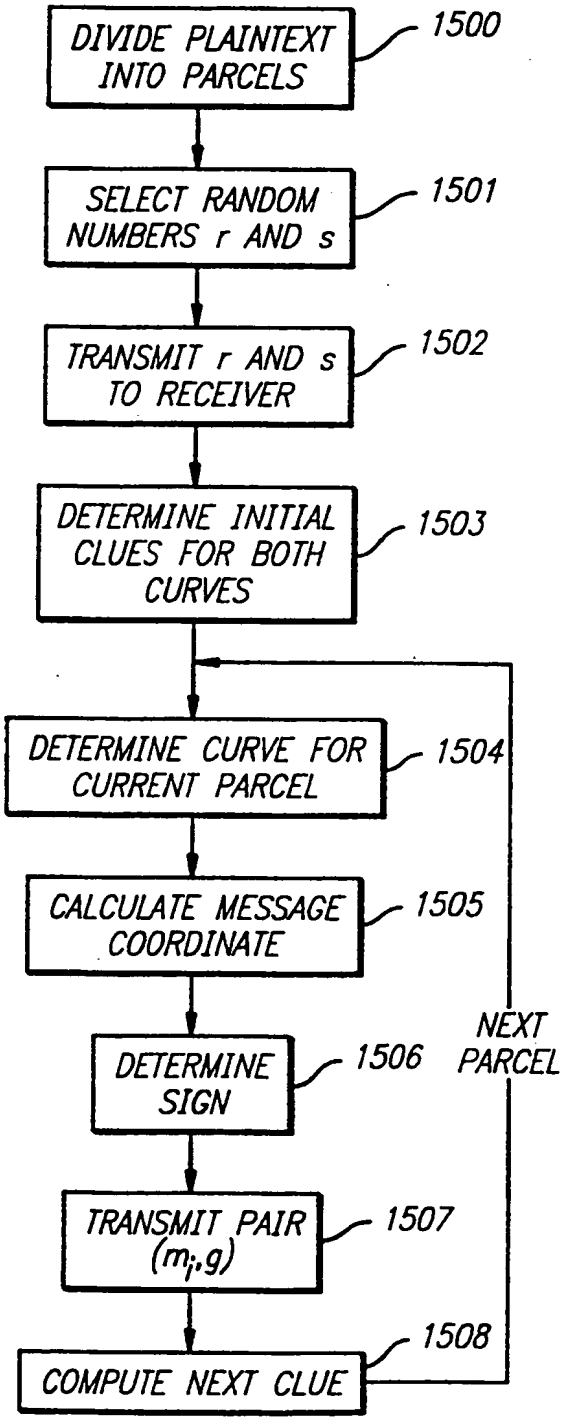
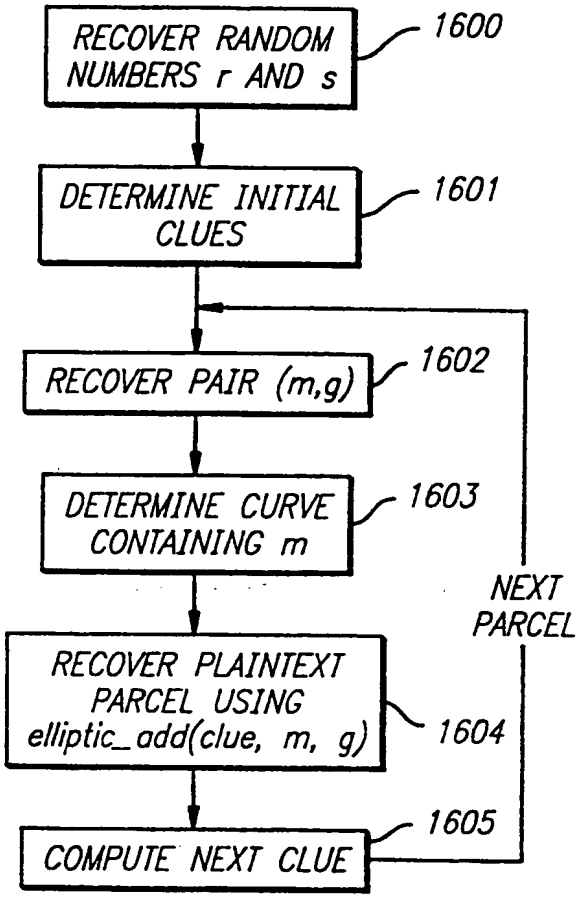


FIG. 16



INTERNATIONAL SEARCH REPORT

Inter nal Application No
PCT/US 98/14892

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 H04L9/30

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5 159 632 A (CRANDALL) 27 October 1992 cited in the application see abstract see column 7, line 1 - line 50 see column 13, line 56 - column 14, line 46	1,9,17, 26
A	US 5 497 423 A (MIYAJI) 5 March 1996 see column 4, line 31 - column 5, line 3	1,9,17, 26

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
27 November 1998	03/12/1998

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Holper, G

1

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

...formation on patent family members

Inter... nal Application No
PCT/US 98/14892

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5159632 A	27-10-1992	AU 2697792 A	27-04-1993
		WO 9306672 A	01-04-1993
		US 5463690 A	31-10-1995
		US 5581616 A	03-12-1996
		US 5805703 A	08-09-1998
		US 5271061 A	14-12-1993
US 5497423 A	05-03-1996	JP 7098563 A	11-04-1995

Form PCT/ISA/210 (patent family annex) (July 1992)